Broadview
www.broadview.com.cn

MATLAB
精品丛书

MATLAB/Simulink

通信系统建模与仿真实例精讲

邵佳 董辰辉 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Broadview®
www.broadview.com.cn
博文视点 · IT出版旗舰品牌
技术凝聚实力 · 专业创新出版

MATLAB/Simulink

通信系统建模与仿真实例精讲

本书深入浅出地介绍了Simulink在通信系统方面的建模与仿真设计技术，知识全面、技术新颖，涉及了许多前沿通信成果。

本书讲练结合，突出应用实践，5个通信系统仿真综合实例典型实用、全部取自于一线工程实践，利于读者学习后举一反三。

本书盘书结合，光盘中附有实例的仿真文件和程序代码，读者稍加修改，便可应用于自己的工作中或者完成自己的课题，物超所值。

本书导读图

MATLAB/Simulink 基础技术篇

- ◎通信系统与仿真专业基础
- ◎MATLAB/Simulink仿真原理与操作

通信系统常用模块仿真篇

- ◎信号与信道
- ◎调制与解调
- ◎通信滤波器
- ◎同步
- ◎信源编码/译码
- ◎均衡器与射频损耗
- ◎差错控制编码/译码

通信系统仿真综合实例篇

- ◎蓝牙跳频通信系统仿真设计
- ◎直接序列扩频通信系统仿真设计
- ◎IS-95前向链路通信系统仿真设计
- ◎OFDM通信系统仿真设计
- ◎MIMO通信系统仿真设计

图书分类：计算机>MATLAB

ISBN 978-7-121-08777-6

9 787121 087776 >

定价：49.00元（含光盘1张）

责任编辑：江　立
责任美编：李　玲

MATLAB
精品丛书

MATLAB/Simulink

通信系统建模与仿真实例精讲

邵佳 董辰辉 编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

全书以实际工程为背景，通过专业技术与大量实例结合的形式，系统详细地介绍了 MATLAB/Simulink 2008 通信系统建模与仿真设计的方法和技巧。全书共分 3 篇 14 章，第 1～2 章为 MATLAB/Simulink 基础技术篇，简要介绍了通信系统基础知识、集成环境 MATLAB/Simulink、S-function 设计与应用；第 3～9 章为通信系统常用模块仿真篇，重点对信号与信道、信源编码/译码、调制与解调、均衡器与射频损耗、通信滤波器、差错控制编码/译码、同步与其他模块的建模与仿真技术进行了阐述；第 10～14 章为通信系统仿真综合实例篇，深入浅出地剖析了蓝牙跳频通信系统、直接序列扩频通信系统、IS-95 前向链路通信系统、OFDM 通信系统以及 MIMO 通信系统建模与仿真设计的流程和细节。这 5 个工程案例典型实用，技术前沿新颖，代表了通信系统的先进成果。读者通过学习，将可以举一反三，快速提高应用水平，胜任各种 MATLAB/Simulink 通信系统的建模与仿真设计工作。

本书配有光盘 1 张，包含了全书所有实例的硬件原理图和程序源代码，方便读者学习和使用。本书适合信息与通信工程等相关专业的大学生，以及从事 MATLAB/Simulink 仿真的科研人员使用。

图书在版编目（CIP）数据

MATLAB/Simulink 通信系统建模与仿真实例精讲 / 邵佳，董辰辉编著. —北京：电子工业出版社，2009.6
（MATLAB 精品丛书）
ISBN 978-7-121-08777-6

I. M… II.①邵…②董… III.①通信系统－系统建模－计算机辅助计算－软件包，Matlab、Simulink②通信系统－系统仿真－计算机辅助计算－软件包，Matlab、Simulink IV.TN914-39

中国版本图书馆 CIP 数据核字（2009）第 071368 号

责任编辑：江　立
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：22.75　字数：508 千字
印　　次：2009 年 6 月第 1 次印刷
印　　数：3500 册　　定价：49.00 元 （含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

目前网络通信是一个非常热门的领域，无论是有线网络还是无线网络，都逐渐渗透应用到生活的各个地方，通信系统正向着宽带化方向迅速发展；而 Simulink 是 MATLAB 中的一种可视化仿真工具，广泛应用于线性系统、数字控制、非线性系统以及数字信号处理的建模和仿真中。使用 MATLAB/Simulink 进行通信系统建模与仿真设计，已经成为大量通信工程师必须研究掌握的技术之一，也是高等院校通信专业学生学习的课程之一。

但目前市场上同类 MATLAB/Simulink 通信系统仿真设计的书非常少，仅有的几本也以介绍基础理论为主，实用性不强，缺少通过大量实例讲解 MATLAB/Simulink 通信系统建模与仿真设计的内容，远远满足不了广大读者学习的需求。本书就是为了弥补这种不足而编写的。

本书内容

全书以实际工程为背景，重点通过专业技术与大量实例结合的形式，详细地介绍了 MATLAB/Simulink 2008 通信系统建模与仿真设计的方法和技巧。全书共分 3 篇 14 章，具体内容安排如下。

第一篇为 MATLAB/Simulink 基础技术篇，包括第 1～2 章，简要介绍了通信系统与仿真基础知识、集成环境 MATLAB/Simulink、S-function 设计与应用，引导读者进行 MATLAB/Simulink 入门。

第二篇为通信系统常用模块仿真篇，包括第 3～9 章，内容包括信号与信道、信源编码/译码、调制与解调、均衡器与射频损耗、通信滤波器、差错控制编码/译码、同步与其他模块的仿真。读者通过学习，将对通信系统常用模块与仿真技术有所熟悉，并了解相关的一些简单性应用。

第三篇为通信系统仿真综合实例篇，包括第 10～14 章，通过蓝牙跳频通信系统、直接序列扩频通信系统、IS-95 前向链路通信系统、OFDM 通信系统以及 MIMO 通信系统，深入浅出地剖析了 MATLAB/Simulink 通信系统仿真设计流程及应用方法。读者通过学习，设计水平将快速提高，迅速实现入门到精通的飞跃。

本书配有光盘一张，包含了全书所有实例的硬件原理图和程序源代码，方便读者学习和使用。本书适合信息与通信工程等相关专业的大学生，以及从事 MATLAB/Simulink 仿真的科研人员使用。

本书特色

与同类书比较，本书主要特点如下：

（1）以 MATLAB/Simulink 2008 最新版为蓝本，深入介绍了 Simulink 通信系统的建模

与仿真设计技术，知识系统、技术新颖，反映了许多先进通信成果。

（2）本书讲练结合，突出应用实践，5 个通信系统综合实例典型实用，全部取自于一线工程实践，利于读者学习后举一反三。

（3）本书盘书结合，光盘中附有实例的仿真文件和程序代码，读者稍加修改，便可应用于自己的工作中或者完成自己的课题，物超所值。

本书作者

本书主要由邵佳、董辰辉编著。另外参加编写的人还有：唐清善、邱宝良、周克足、刘斌、李永怀、李宁宇、黄小欢、严剑忠、黄小宽、李彦超、付军鹏、张广安、王艳波、金平、徐春林、谢正义、郑贞平、张小红等。他们在资料收集、整理和技术支持方面做了大量的工作，在此一并向他们表示感谢！

由于时间仓促，再加之作者的水平有限，书中难免存在一些不足之处，欢迎广大读者批评指正。

目　录

第一篇　MATLAB/Simulink 基础技术篇

第二篇　通信系统常用模块仿真篇

第三篇　通信系统仿真综合实例篇

Part 1

MATLAB/Simulink 基础技术篇

第 1 章　通信系统与仿真专业基础

人类社会要进行信息交流就离不开通信，通信按照传统的理解就是信息的传输与交换。现在社会已进入信息时代，人们要进行交换的信息已不再是单一的电话业务，而是集声、图、文为一体的综合性的多种信息服务，因此现在通信网是一个综合业务数字网。本书将系统介绍 MATLAB/Simulink 通信系统仿真技术与实例，首先介绍通信系统与仿真的基础知识。

1.1 通信系统概述

通信系统（Communication System）是用以完成信息传输过程的技术系统的总称。现代通信系统主要借助电磁波在自由空间的传播或在导引媒体中的传输机理来实现，前者称为无线通信系统，后者称为有线通信系统。由于人们对通信的容量要求越来越高，对通信的业务要求越来越多样化，所以通信系统正迅速向着宽带化方向发展，而光纤通信系统将在通信网中发挥越来越重要的作用。

1.2 通信系统的组成

通信是将信息从发信者传递给在另一个时空点的收信者。由于完成这一信息传递的通信系统的种类繁多，因此它们的具体设备和业务功能可能各不相同，经过抽象概括，通信流程可用如图 1-1 所示的基本模型图来表示。整个流程是由信源、发送设备、信道（或传输媒质）、接收设备和收信者（信宿）五部分组成。

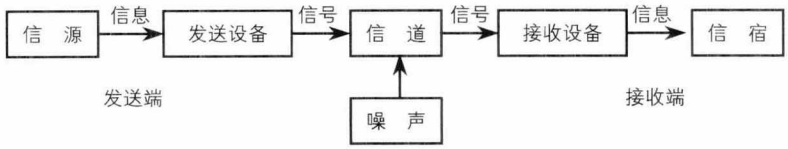


图 1-1　通信系统的基本模型

上述模型概括地反映了通信系统的共性。根据我们的研究对象及所关心的不同问题，将会使用不同形式的较具体的通信系统模型。

1.2.1　信源

信源是信息的产生者或信息的形成者。根据信源所产生信号的性质不同，可分为模拟信源和离散信源。

模拟信源（如电话机和电视摄像机等）输出幅度连续的信号；离散信源（如电传机、计算机等）输出离散的符号序列或文字。模拟信源可通过抽样和量化转换为离散信源。随着计算机和数字通信技术的发展，离散信源的种类和数量会越来越多。这里需要强调指出，随着信源和接收者的不同，信息的速率将在很大范围内变化。例如，一台电传打字机的速率为 50bit/s，而彩色电视的速率为 270Mbit/s。由于信源产生的种类和速率不同，因而对传输系统的要求也各不相同。

1.2.2　发送设备

发送设备的基本功能是将信源和传输媒介匹配起来，即将信源产生的消息信号变换为有利于传送的信号形式送往传输媒介。变换方式是多种多样的，在需要频率搬移时，调制是最常见的变换方式。发送设备还包括为达到某些特殊要求所进行的各种处理，如多路复用、保密处理和纠错编码处理等。

1.2.3　信道

信道是指信号传输的媒介，信号是经过信道传送到接收设备的。传输媒介既可以是有线的，也可以是无线的，二者都有多种物理传输媒介。

在信号传输过程中，必然会引入发送设备、接受设备和传输媒介的热噪声和各种干扰和衰减，即信号在信道中传输时，会产生信道噪声。

媒介的固有特性和干扰特性会直接影响变换方式的选取，如通过电导体传播的有线信道和通过自由空间传播的无线信道，其信号变换方式是不同的。不同频段的无线电波在空间传播的途径、性能和衰减（衰落）也是不同的。

1.2.4　接收设备

接收设备的主要作用是将来自信道的带有干扰的发送信号加以处理，并从中提取原始信息，完成发送变换过程的逆变换——解调和译码。对于多路复用信号，还包括多路去复用，实现正确分路。由于接收的消息信号存在噪声和传输损伤，接收设备还可能包含趋近理想恢复的某些措施和方法。

1.2.5　信宿

信宿是将复原的原始信号转换成相应的消息。

应当指出，上述模型是点对点的单向通信系统。对于双向通信，通信双方都要有发送设备和接收设备。对于多个用户之间的双向通信，为了能实现信息的有效传输，必须进行信息的交换和分发，由传输系统和交换系统组成的一个完整的通信系统或通信网络来实现。其中交换系统完成不同地址信息的交换，因此交换系统中的每一台交换机组成了通信网络中的各个节点。

一个实际的通信系统往往由终端设备、传输链路和交换设备三大部分组成。

（1）终端设备

终端设备主要功能是把待传送的信息与在信道上传送的信号相互转换。这就要求有发送传感器和接收传感器将信号恢复成能被利用的信息，还应该有处理信号的设备以便能与信道匹配。另外，还需要有能产生和识别通信系统内所需的信令信号或规约。对应不同的电信业务有不同的信源和信宿，也就有着不同的变换的反变换设备，因此对应不同的电信业务也就有不同的终端设备，如电话业务的终端设备就是电话机，传真业务的终端设备就是传真机，数据业务的终端设备就是数据终端机等。

（2）传输链路

传输链路是连接源点和终点的媒介和通路，除对应于通信系统模型中信道部分之外，还包括一部分变换和反变换设备。

传输链路的实现方式主要有以下几种：

- 物理传输媒介本身就是传输链路，如实线和电缆；
- 采用传输设备和物理传输媒介一起形成的传输链路，如载波电路和光通信链路；
- 传输设备利用大气传输链路，如微波和卫星通信链路。

（3）交换设备

交换设备是现代通信网的核心，其基本功能是完成接入交换节点链路的汇集、转换和分配。对不同电信业务网络的转接，交换设备的性能要求也不相同。例如，电话业务网的交换设备实时性强，因此目前电话业务网主要采用直接接续通话电路的交换方式。

对于主要用于计算机通信的数据业务网，由于数据终端或计算机可有各种不同的速率，为了提高链路利用率，可将流入信息流进行分组、存储，然后再转发到所需链路上去，这种方式叫做分组交换方式。例如，分组数据交换机就按这种方式进行交换，这种方式可以比较高效地利用传输链路。

1.3 通信系统的分类

通信系统的分类方法很多，既可以按用途来分，也可以按传输信号的特征来分，还可以按工作方式来分。本节仅对图 1-1 所示的通信系统模型所引出的分类方法进行讨论。

1.3.1 按信源分类

按照信源发出消息的物理特征不同可分为电话、电报、数据和图像等通信系统。其中电话通信目前最发达，其他通信常借助于公共电话通信系统传递信息。如电报通信一般采用公共电话系统中的一个话路或从话路中的一部分频带进行传送；电视信号或图像信号可使用多个话路合并为一个信道进行传送。

1.3.2 按传输媒介分类

通信系统模型中的信道是指传输信息的媒介或信号的通道。按传输媒介分类，通信系

统可分为有线（包括光纤）和无线两大类。表 1-1 列出了常用的媒介及其用途。

表 1-1　常用传输媒介及其用途

频率范围	波长范围	表示符号	传输媒介	主要用途或场合
3Hz~30kHz	10^8~10^4m	VLF（甚低频）	有线线对（超长波）	音频、电话、数据终端
30~300kHz	10^4~10^3m	LF（低频）	有线线对（长波）	导航、信标、电力线、通信
300kHz~3MHz	10^3~10^2m	MF（中频）	同轴电缆（中波）	AM 广播、业余无线电
3~30MHz	10^2~10m	HF（高频）	同轴电缆（短波）	移动电话、短波广播、业余无线电
30~300MHz	10~1m	VHF（甚高频）	同轴电缆（米波）	FM 广播、TV、导航、移动通信
300M~3GHz	1m~10cm	UHF（特高频）	同轴电缆、 波导（分米波）	TV、遥控遥测、雷达、移动通信
3~30GHz	10~1cm	SHF（超高频）	波导（厘米波）	微波通信、卫星通信、雷达
30~300GHz	10~1mm	EHF（极高频）	波导（毫米波）	微波通信、雷达、射电天文学

由表 1-1 中的信道可构成相应的通信系统。有线信道常用的是对称电缆、同轴电缆和光缆，由此可构成电缆通信系统和光纤通信系统。目前国际和我国长途通信系统中主要采用的是光纤通信系统，而电缆通信系统大都用在本地通信系统中。无线信道按照所使用的频段和通信手段可分为短波通信系统、微波中继通信系统、移动通信系统和卫星通信系统。

1.3.3　按传输信号的特征分类

根据传输信号的特征，通信系统可以分为模拟通信系统和数字通信系统两大类。

（1）模拟通信系统

在模拟通信系统中传输的是模拟信号。如图 1-2 所示的是模拟通信系统的基本组成。在图中用调制器取代图 1-1 中的发送变换器，用解调器取代了图 1-1 中的接收变换器。这里的调制器和解调器对信号的变换起着决定性的作用，直接关系着通信质量的优劣。

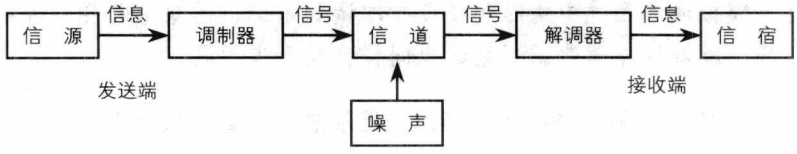


图 1-2　模拟通信系统的基本组成图

（2）数字通信系统

在数字通信系统中传输的是数字信号。数字通信系统的基本组成如图 1-3 所示。数字通信系统除了包括调制器和解调器外，还包括信源编码器、信道编码器、信道译码器、信源译码器和同步系统等。

数字通信系统的组成主要有以下几个部分：

- 信源编码器

信源编码器的主要作用是提高数字信号传输的有效性。如果信息源是数据处理设备，

还要进行并/串变换，以便进行数据传输。通常的数字加密也可归并到信源编码中。接收端的信源译码是信源编码的逆变换。

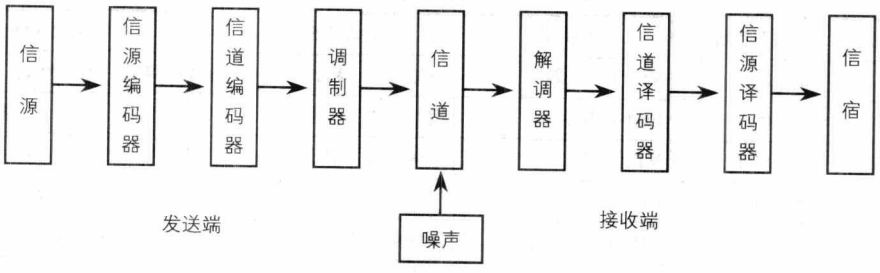


图 1-3　数字通信系统的基本组成图

- 信道编码器

信道编码器主要是为了提高数字信号传输的可靠性。由于传输信道内噪声的存在和信道特性不理想造成的码元间干扰，通信系统很容易产生传输差错，而信道的线性畸变所造成的码间干扰可通过均衡办法基本消除，因此信道中的噪声是导致传输差错的主要原因。减小这种差错的基本做法是在信码组中按一定规则附加上若干个监视码元（或称冗余度码元），使原来不相关的数字信息序列变为相关的新序列，然后在接收端根据这种相关的规律性来检测或纠正接收序列码组中的误码，提高可靠性，因此信道编码器又称为差错控制编码器。接收端的信道译码器是信道编码器的逆过程。

- 同步系统

同步系统用于建立通信系统收、发相对一致的时间关系。只有这样，接收端才能确定每位码的起止时间，并确定接收码组与发送码组的正确对应关系，否则接收端无法恢复发端的信息。因此同步是数字通信系统正常工作的前提，通信系统能否有效地、可靠地工作，很大程度上依赖于同步系统性能的好坏。同步可分为载波同步、位同步、帧同步和网同步四大类。

对于模拟通信系统中的时分多路脉冲调制系统、图像（电视）传输系统和采用相干解调的连续波调制系统也同样存在同步问题。

模拟通信系统与数字通信系统各有特点，但从总体上看，数字通信系统与模拟通信系统相比，具有以下优点：

- 抗干扰能力强，数字通信系统可通过再生中继器消除噪声积累；
- 可采用差错控制技术，从而提高数字信号传输的可靠性；
- 便于进行各种数字信号处理，如计算机存储和处理，使数字通信和计算机技术相结合而组成综合化、智能化的数字通信网；
- 数字通信系统可使传输与交换相结合，电话、数据和图像传输相结合，有利于实现综合业务数字网；
- 数字通信系统的器件和设备易于实现集成化、微型化。

然而数字通信系统也存在占用频带宽的缺点，但近年来卫星通信和光纤通信等宽带通

信系统日趋发展成熟，为数字通信系统提供了足够宽的频带，因而相比之下，此缺点就不显得突出了。

1.4 仿真技术与通信仿真

仿真是衡量系统性能的工具，它通过仿真模型的仿真结果来推断原系统的性能，从而为新系统的建立和原系统的改造提供可靠的参考。仿真是科学研究和工程建设中不可缺少的方法。

实际的通信系统是一个功能结构相当复杂的系统，对这个系统做出的任何改变都可能影响到整个系统的性能和稳定。因此，在对原有的通信系统做出改进或建立一个新系统之前，通常对这个系统进行建模和仿真，通过仿真结果衡量方案的可行性，从中选择最合理的系统配置和参数设置，然后再应用到实际系统中，这个过程就是通信仿真。

1.4.1 仿真技术

仿真技术是以相似原理、系统技术、信息技术以及仿真应用领域的有关技术为基础，以计算机系统、与应用有关的物理效应设备及仿真器为工具，利用模型对系统（已有的或设想的）进行研究的一门多学科的综合性技术。

仿真本质上是一种知识处理的过程。典型的系统仿真过程包括：系统模型建立、仿真模型建立、仿真程序设计、模型确认、仿真实验和数据分析处理等，它涉及很多领域的知识和经验。系统仿真可以有很多种分类方法。按模型的类型可以分为连续系统仿真、离散系统仿真、连续/离散（时间）混合系统仿真和定性系统仿真；按仿真的实现方法和手段可以分为物理仿真、计算机仿真、硬件在回路中的仿真（半实物仿真）和人在回路中的仿真；根据人和设备的真实程度，可以分为实况仿真、虚拟仿真和构造仿真。

1.4.2 计算机仿真的一般过程

前面提到过，仿真在实现方法上可以分为多种。而本书介绍的 Simulink 的仿真技术则属于计算机仿真的一种。计算机仿真的一般过程可以表述如下。

（1）描述仿真问题，明确仿真目的。

（2）项目计划、方案设计与系统定义。

根据仿真相应的结构，规定相应仿真系统的边界条件与约束条件。

（3）数学建模。

根据系统的先验知识、实验数据及其机理研究，按照物理原理或者采取系统辨识的方法，确定模型的类型、结构及参数。注意，要确保模型的有效性和经济性。

（4）仿真建模。

根据数学模型的形式、计算机类型、采用的高级语言或其他仿真工具，将数学模型转换成能在计算机上运行的程序或其他模型。

（5）实验。

设定实验环境/条件和记录数据，进行实验并记录数据。

（6）仿真结果分析。

根据实验要求和仿真目的对实验结果进行分析处理，根据分析结果修正数学模型、仿真模型、仿真程序或者修正/改变原型系统，以进行新的实验。模型是否能够正确地表示实际系统，并不是一次完成的，而是需要比较模型和实际系统的差异，不断地修正和验证才能完成。

1.4.3 通信仿真的概念

通信仿真是衡量通信系统性能的工具。通信仿真可以分为离散事件仿真和连续仿真。在离散事件仿真中，仿真系统只对离散事件做出响应；而在连续仿真中，仿真系统对输入信号产生连续的输出信号。离散事件仿真是对实际通信系统的一种简化，它的仿真建模比较简单，整个仿真过程需要花费的时间也比连续仿真少。虽然离散事件仿真舍弃了一些仿真细节，在有些场合显得不够具体，但仍然是通信仿真的主要形式。

与一般的仿真过程类似，在对通信系统实施仿真之前，首先需要研究通信系统的特性，通过归纳和抽象建立通信系统的仿真模型。通信系统仿真是一个循环往复的过程，它从当前系统出发，通过分析建立起一个能够在一定程度上描述原通信系统的仿真模型，然后通过仿真实验得到相关的数据。通过对仿真数据的分析可以得到相应的结论，然后把这个结论应用到对当前通信系统的改造中。如果改造后通信系统的性能并不像仿真结果那样令人满意，还需要重新实施通信系统仿真，这时候改造后的通信系统就成了当前系统，并且开始新一轮的通信系统仿真过程。

值得注意的是，在整个通信系统的仿真过程中，人为因素自始至终起着相当重要的作用。除了仿真程序的运行外，通信仿真的每个步骤都需要进行人工干预，由人对当前的情况做出正确的判断。因此，通信仿真并不是一个机械的过程，它实际上是人的思维活动在计算机协助下的一种延伸。

1.4.4 通信仿真的一般步骤

通信系统仿真一般分为 3 个步骤，即仿真建模、仿真实验和仿真分析。应该注意的是，通信仿真是一个螺旋式发展的过程，因此，这 3 个步骤可能需要循环执行多次之后才能够获得令人满意的仿真结果。

（1）仿真建模

仿真建模是根据实际通信系统建立仿真模型的过程，它是整个通信仿真过程中的一个关键步骤，因为仿真模型的好坏直接影响着仿真的结果以及仿真结构的真实性和可靠性。

仿真模型是对实际系统的一种模拟和抽象。过于简单的仿真模型会忽略实际系统的细节，在一定程度上会影响仿真结果的可靠性。但过于复杂的仿真模型则会产生很多相互因素，从而大大延长仿真时间和增加仿真结果分析的复杂度。因此，仿真模型的建立需要综

合考虑其可行性和简单性。在仿真建模过程中，可以先建立一个相对简单的仿真模型，然后再根据仿真结果和仿真过程的需要逐步增加仿真模型的复杂度。

在仿真建模过程中，首先需要分析实际系统存在的问题或设立系统改造的目标，并把这些问题和目标转化成数学变量和公式。确定了仿真目标后，下一步是获取实际通信系统的各种运行参数，如通信系统占用的带宽及其频率分布、系统对于特定的输入信号产生的输出等。

在以上工作准备好以后，就是仿真软件的选择了。除了使用传统的编程语言外，目前工程技术人员比较倾向于更加专业和方便使用的专门的仿真软件。比较常见的包括MATLAB、OPNET 和 NS2 等。

使用仿真软件建立好仿真模型后，仿真建模的这一步骤就基本完成了。值得注意的是，在进行下一步工作前，要做好仿真模型文档说明，这有利于使仿真工作条例更加清晰，在调试过程中能够很容易找出错误所在并及时纠正。

（2）仿真实验

仿真实验是一个或一系列针对仿真模型的测试。在仿真实验过程中，通常需要多次改变仿真模型输入信号的数值，以观察和分析仿真模型对这些输入信号的反应，以及仿真系统在这个过程中表现出来的性能。需要强调的一点是，仿真过程中使用的输入数据必须具有一定的代表性，即能够从各种角度显著地改变仿真输出信号的数值。

在明确了仿真系统对输入/输出信号的要求之后，最好把这些设置整理成一份简单的文档。编写文档是一个好习惯，它能够帮助回忆起仿真设计过程的一些细节。当然，文档的编写不一定要求很规范，并且文档大小应该视仿真设计的规模而定。

对于需要较长时间的仿真，应该尽可能地使用批处理方式，使得仿真过程在完成一种参数配置的仿真之后，能够自动启动针对下一个仿真参数配置的下一次仿真。这种方式减少了仿真过程中的人工干预，提高了系统利用率和仿真效率。

（3）仿真分析

仿真分析是一个通信仿真流程的最后一个步骤。在仿真分析过程中，用户已经从仿真过程中获得了足够多的关于系统性能的信息，但是这些信息只是一些原始数据，一般还需要经过数值分析和处理才能够获得衡量系统性能的尺度，从而获得对仿真性能的一个总体评价。常用的系统性能尺度包括平均值、方差、标准差、最大值和最小值等，它们从不同的角度描绘了仿真系统的性能。

需要注意的是，即使仿真过程中收集的数据正确无误，由此得到的仿真结果并不一定就是准确的。造成这种结果的原因可能是输入信号恰好与仿真系统的内部特性吻合，或者输入的随机信号不具有足够的代表性。

图表是最简洁的说明工具，它们具有很强的直观性，便于分析和比较，因此，仿真分析的结果一般都制成图表形式。而且，一般使用的仿真工具都具有很强的绘图功能，能够便捷地绘制各种类型的图表。

以上就是通信系统的一个循环。应该强调的是，仿真分析并不一定意味着通信仿真过

程的完全结束。如果仿真分析得到的结果达不到预期的目标，用户还需要重新修改通信仿真模型，这时候仿真分析就成为了一个新循环的开始。

1.5 本章小结

本章介绍了通信系统、仿真技术与通信仿真的基本概念，说明了仿真的基本步骤和应该注意的问题。通过本章的学习，读者将对通信系统与仿真技术有一个入门性的了解，下面章节将介绍 MATLAB/Simulink 通信环境与操作。

第 2 章　MATLAB/Simulink 仿真原理与操作

本章将介绍 Simulink 仿真环境以及一些基本操作。首先简单介绍 MATLAB/Simulink 特点及其工作原理。

2.1 MATLAB/Simulink 特点及工作原理

Simulink 是 MATLAB 中的一种可视化仿真工具，广泛应用于线性系统、数字控制、非线性系统以及数字信号处理的建模和仿真中。Simulink 采用模块化建模方式，每个模块都有自己的输入/输出端口，实现一定的功能。

2.1.1 Simulink 主要特点

Simulink 是 MATLAB 提供的用于对动态系统进行建模、仿真和分析的工具包。Simulink 提供了专门用于显示输出信号的模块，可以在仿真过程中随时观察仿真结果。同时，通过 Simulink 的存储模块，仿真数据可以方便地以各种形式保存到工作空间或文件中，以供用户在仿真结束之后对数据进行分析和处理。另外，Simulink 把具有特定功能的代码组织成模块的方式，并且这些模块可以组织成具有等级结构的子系统，因此具有内在的模块化设计要求。基于以上优点，Simulink 作为一种通用的仿真建模工具，广泛用于通信仿真、数字信号处理、模糊逻辑、神经网络、机械控制和虚拟现实等领域中。

作为一款专业仿真软件，Simulink 具有以下特点：

- 基于矩阵的数值计算；
- 高级编程语言以及可视化的图形操作界面；
- 包含各个领域的仿真工具箱，使用方便快捷并可以扩展；
- 丰富的数据 I/O 接口；
- 提供与其他高级语言的接口；
- 支持多平台（PC/UNIX）。

根据输出信号与输入信号的关系，Simulink 提供 3 种类型的模块：连续模块、离散模块和混合模块。连续模块是指输出信号随着输入信号发生连续变化的模块；离散模块则是输出信号以固定间隔变化的模块。对于连续模块，Simulink 采用积分方式计算输出信号的数值，因此，连续模块主要涉及数值的计算及其积分。离散模块的输出信号在下一个抽样到来之前保持恒定，这时候，Simulink 只需以一定的间隔计算输出信号的数值。混合模块是根据输入信号的类型来确定输出信号类型的，它既能够产生连续输出信号，也能够产生

离散输出信号。

如果一个仿真模型中只包含离散模块，这时候，Simulink 采用固定步长的方式进行仿真（即每隔一定的间隔计算一次输出信号）。当所有的离散模块都有相同的抽样间隔时，Simulink 只需要按照这个间隔实施仿真；否则，Simulink 采用多速率方式进行仿真。多速率仿真模式的一种方案是选取一个最大可用间隔，使之适用于所有的离散模块。这个间隔一般是各个离散模块抽样间隔的最大公约数。对于可变步长方式，多速率仿真模型按照各个模块的抽样间隔列出系统可能的仿真时刻，在仿真时刻到来的时候，只对相应的离散模块实施仿真，从而在一定程度上提高了仿真的效率。

如果仿真模型中包含了连续模块，Simulink 将采用连续方式对模块进行仿真。如果模块中既包括连续模块，又包含离散模块，Simulink 采用两种仿真步长进行仿真。对于其中的离散模块，Simulink 可以按照离散模块的方式进行仿真，这个仿真步长称为主步长。在每个步长仿真中，Simulink 使用小步长间隔，通过积分运算得到连续状态的当前输出信号。

2.1.2 Simulink 仿真的工作原理

Simulink 仿真包括两个阶段：初始化阶段和模型执行阶段。

（1）初始化阶段

在初始化阶段，Simulink 内部主要完成以下工作。

- 模型参数传给 MATLAB 进行估值，得到的数值结果将作为模型的实际参数。
- 展开模型的各个层次，每一个非条件执行的子系统被它包含的模块代替。
- 模型中的模块按更新的次序进行排序。

排序算法产生一个列表，以确保具有代数环的模块在产生它的驱动输入的模块被更新后才更新。当然，这一步要先检测出模型中存在的代数环。

- 决定模型由无显式设定的信号属性。

例如名称、数据类型、数值类型以及大小等，并且检查每个模块是否能够接收连接到它们输入端的信号。Simulink 使用属性传递来确定未被设定的属性，这个过程将源信号的属性传递到它所驱动的模块的输入信号。

- 决定所有无显示设定才采样的时间的模块的采样时间。
- 分配和初始化用于存储每个模块的状态和输入当前值的存储空间。

完成以上工作后，就可以进行下一步工作了，也就是模型执行阶段。

（2）模型执行阶段

一般模型是使用数值积分来进行仿真的，所运用的仿真解法器（仿真算法）依赖于模型提供它的连续积分能力。计算微分可分为以下两步来进行。

首先，按照排列所确定的次序计算每个模块的输出。

然后，根据当前时刻的输入和状态来决定状态的微分；得到微分向量后再把它返回给解法器；后者用它来计算下一个采样点的状态向量。一旦新的状态向量计算完毕，被采样的数据源模块和接收模块才被更新。

在仿真开始时，模型设定待仿真系统的初始状态和输出。在每个时间步中，Simulink 计算系统的输入、状态和输出，并更新模型来反映计算出的值。在仿真结束时，模型得出系统的输入、状态和输出。

在每个时间步中，Simulink 所采取的动作依次如下：

（1）按照安排好的次序更新模型中模块的输出。Simulink 通过调用模块的输出函数计算模块的输出。Simulink 只把当前值、模块的输入以及状态量传给这些函数计算模块的输出。对于离散系统，Simulink 只有在当前时间是模块采样时间的整数倍时，才会更新模块的输出。

（2）按照安排好的次序更新模块中模块的状态。Simulink 计算一个模块的离散状态的方法是调用模块的离散状态更新函数。而对于连续状态，则对连续状态的微分（在模块可调用的函数里，有一个用于计算连续微分的函数）进行数值积分来获得当前的连续状态。

（3）检查模块连续状态的不连续点。Simulink 使用过零检测来检测连续状态的不连续点。

（4）计算下一个仿真时间步的时间。这是通过调用模块获得下一个采样时间函数来完成的。

2.2 Simulink 的常用操作

本节将详细介绍整个模型创建全过程中所涉及的一些操作，包括安装启动、模块基本操作、信号线基本操作、模型的注释和模型的打印等。

2.2.1 安装与启动

（1）安装

安装 Simulink 可以在安装 MATLAB 时随着 MATLAB 一起安装到计算机内。而安装了 MATLAB 却没有安装 Simulink 的话，也可以继续使用安装程序将 Simulink 安装到计算机内。两者方法基本一样。

安装 Simulink 的方法为：启动了 MATLAB 的安装程序后，在 MATLAB 的安装选项对话框中选中 Simulink 组件。其他组件可以按需要选择是否安装。选择后，单击“Next”按钮，计算机就会自动将 Simulink 安装到 MATLAB 的目录下，如图 2-1 所示。

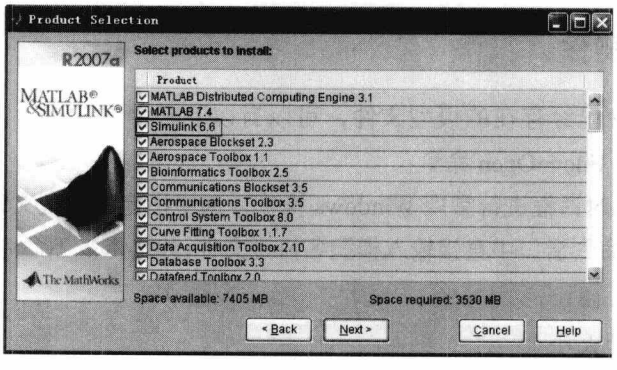


图 2-1　安装 Simulink 组件

不能在不安装 MATLAB 的情况下单独安装 Simulink，因为 Simulink 的运行要依托 MATLAB 来进行。

（2）启动

要启动 Simulink，首先要运行 MATLAB。在 MATLAB 窗口中单击 图标或在命令行中输入如下>> simulink %%%命令启动 Simulink，则会打开 Simulink 库模块浏览器界面，如图 2-2 所示。

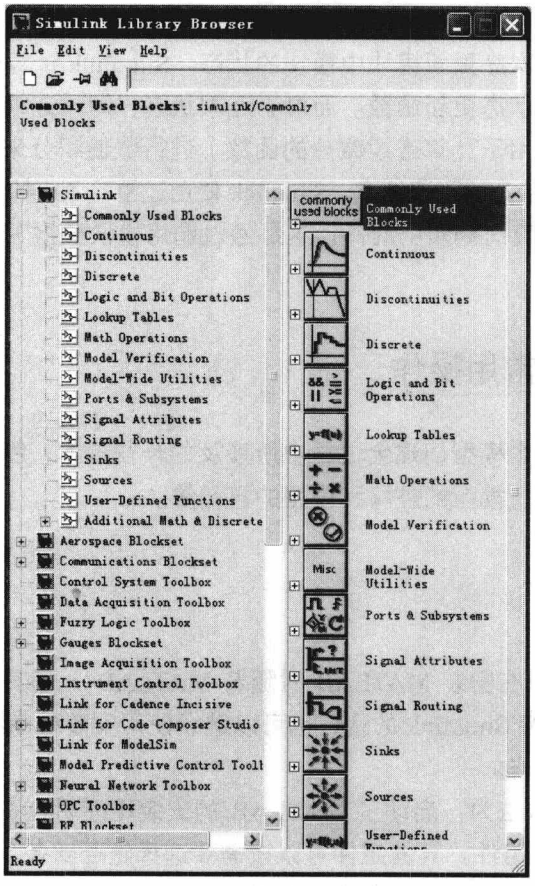


图 2-2　Simulink 库模块浏览器界面

2.2.2　模块基本操作

如需要打开一个已经存在的模型文件，可以有以下三种方法：

- 选择窗口菜单 File→Open 命令；
- 通过对弹出的对话框进行常规 Windows 操作，选中指定的模型文件并且打开；
- 在 MATLAB 命令窗口中直接输入模型名打开文件，如输入以下命令打开 exap.mdl 文件：

```
>> open('exap.mdl')
```

（1）调整模块的大小

在通常情况下，仿真中需要调整某个模块的大小，使得该模块的外观更加适合于建模

的整体设计，看起来更加舒适清晰，增加其可读性。

要实现此操作，具体过程如下。

首先，打开 Random Data Sources 子模块库，如图 2-3 所示。

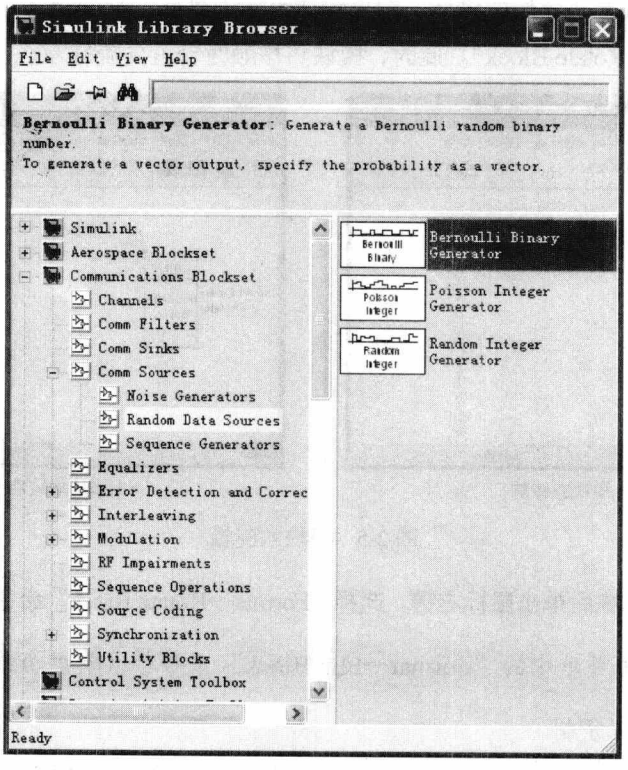


图 2-3　Random Data Sources 子模块库

复制伯努利发生器模块到新建的模型中。在调整其大小之前，首先选中这个模块，此时模块四角出现了 4 个点。单击一个角上的点并且按住鼠标左键，拖动鼠标可实现放大缩小，调整至所需大小后松开鼠标左键即可，如图 2-4 所示。

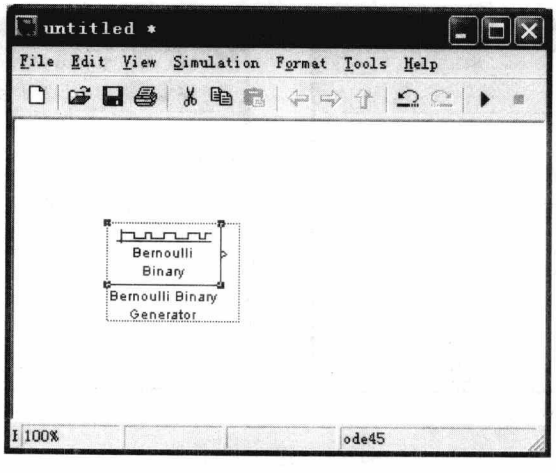
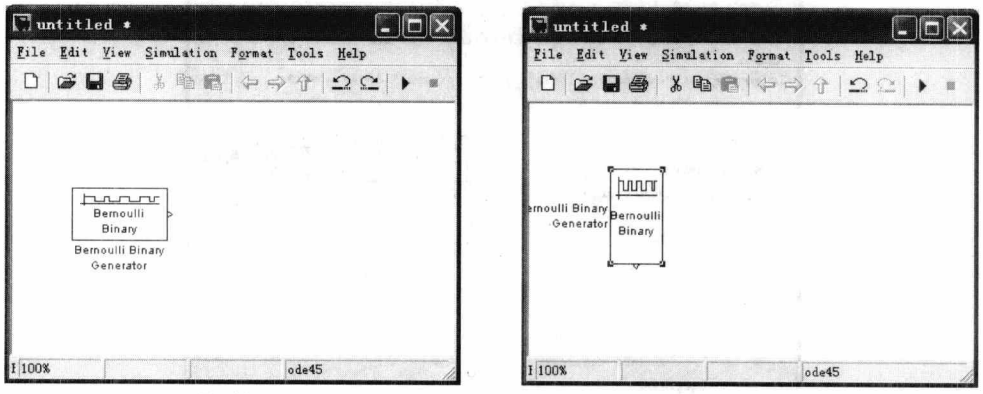


图 2-4　调整伯努利发生器模块大小

在调整时鼠标指针已经改变了形状，并且出现了虚线框以显示调整后的大小。

（2）模块的旋转

有时在建模中需要旋转指定模块，则只需要先选中模块，如图 2-5（a）所示。然后依次单击菜单“Format→Rotate Block”。此时，模块将按顺时针方向旋转 90°，如图 2-5（b）所示。



（a）模块旋转前　　（b）模块旋转后

图 2-5　模块的旋转

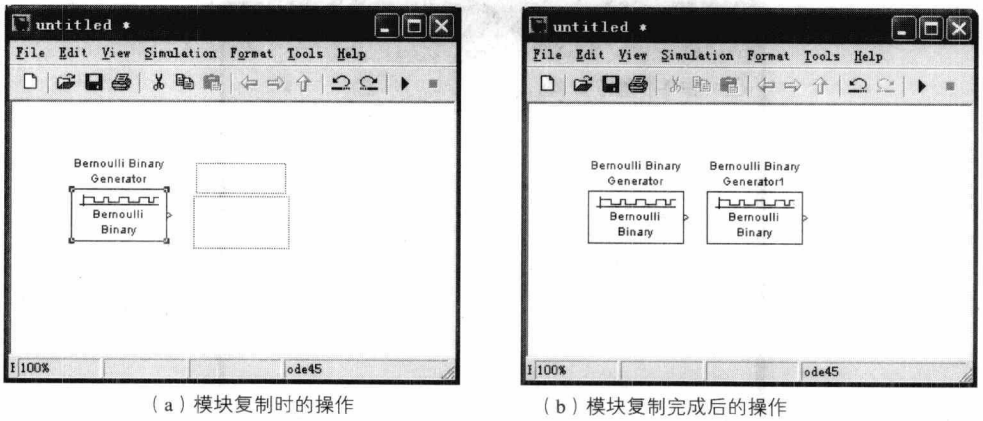
或者可以选中模块后单击鼠标右键，选择“Format→Rotate Block”命令也可以实现此操作。

依次单击菜单中的“Format→Flip Block”命令可以使模块旋转 180°。

（3）模块的内部复制

有时在建模中需要好几个相同的模块，这样就需要用到模块复制。所复制出的模块与原模块具有相同的大小以及相同的属性配置。使用模块的内部复制在许多情况下都使得模型的搭建非常方便。

进行模块的内部复制，首先按住 Ctrl 键，再单击模块，拖动模块到合适的位置即可，如图 2-6（a）所示。放开鼠标左键，则模块内部复制操作就完成了，如图 2-6（b）所示。



（a）模块复制时的操作　　（b）模块复制完成后的操作

图 2-6　模块的复制

当然也可以选中该模块后单击鼠标右键，选择 Copy 命令以及 Paste 命令实现此复制，或者选中模块后单击 图标也可以。

（4）删除模块

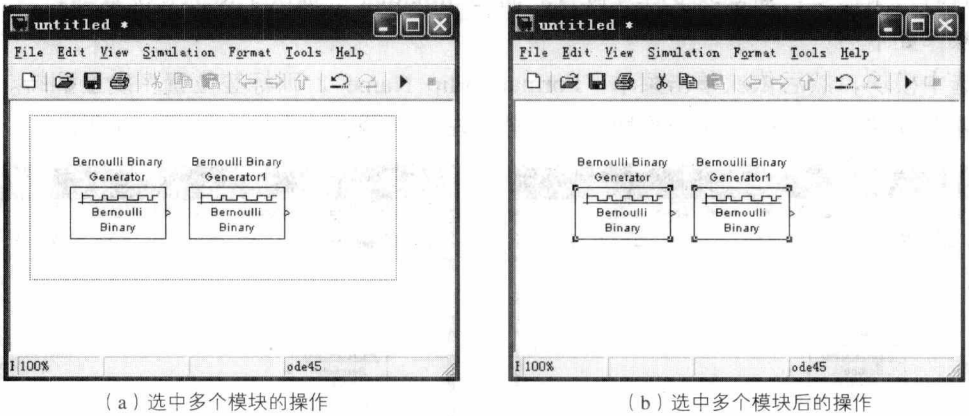
在建模过程中删除模块的操作也是经常用到的，删除模块有以下 4 种方法：

- 选中模块然后按 Delete 键；
- 选中模块然后单击鼠标右键选择 Delete 命令；
- 选中模块然后选择菜单中的“Edit→Cut”命令，可以将模块删除并且保存到剪贴板中；
- 选中模块然后单击 图标，完成“Edit→Cut”相同的功能。

（5）选中多个模块

有时需要同时对多个模块进行相同的操作，如移动、复制和删除等。进行选中多个模块的操作有以下两种方式：

- 按住 Shift 键，然后用鼠标单击想要选中的模块，如果不小心点了不需要的模块，只需要再对其单击一次就可以取消对这个模块的选中；
- 使用范围框，就是按下鼠标左键，然后将所需模块都圈入虚线框内，如图 2-7（a）所示。然后放开鼠标左键即完成操作，如图 2-7（b）所示。



（a）选中多个模块的操作　　（b）选中多个模块后的操作

图 2-7　选中多个模块

如果使用范围框时有模块在范围框外，则按住 Shift 键再单击要选的模块即可，如果在范围框之内有不需要的模块，则按住 Shift 键再单击不需要选的模块即可。

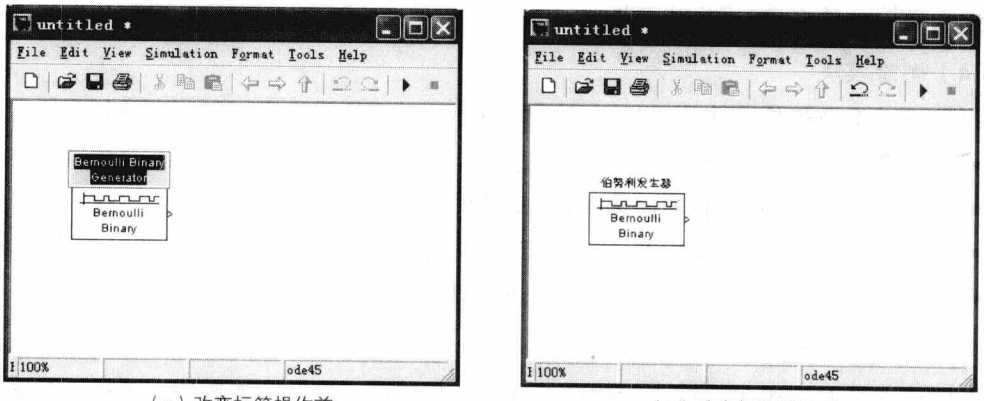
一旦多个模块被同时选中，则可以将这些模块视为一个整体进行复制、删除等操作。

（6）编辑模块的标签

当在模型窗口中创建一个模块时，Simulink 会在每个模块下面的默认位置上加上一个标签。例如，第一个在模型窗口中创建的伯努利发生器模块默认的命名为“Bernoulli Binary Generator”，而第二个则被命名为“Bernoulli Binary Generator1”。不过这个标签是可以根据用户需要改变的。

首先在标签的任何位置双击鼠标左键，则模块标签呈编辑状态，如图 2-8（a）所示；

输入用户自己想要的标签，如图 2-8（b）所示。



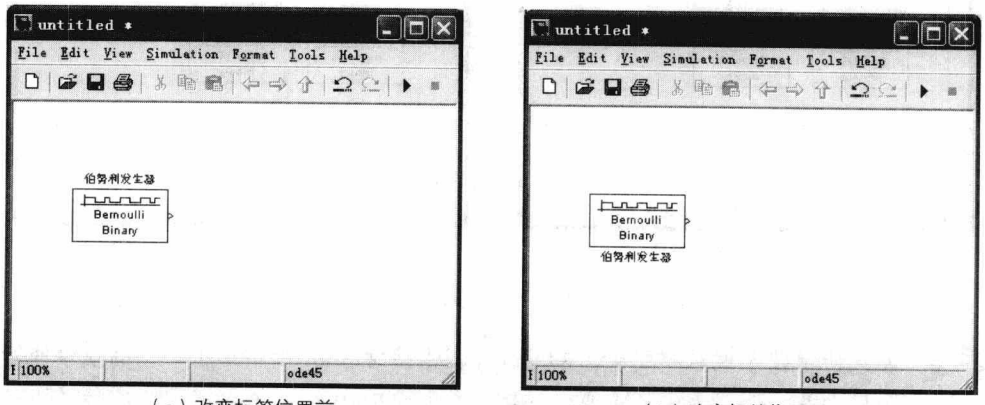
（a）改变标签操作前　　（b）改变标签操作后

图 2-8　改变模块标签

当然，同 Windows 桌面的图标标签的定义相同，还可以多行文字。在标签外的任何位置单击鼠标左键，则新的合法标签将被承认，此模块将由此标签的文字命名。不过需要提醒读者，每个模块的名字至少有一个字符才算合法。

有时为了需要，需要改变标签位置。而在 Simulink 中提供了使标签位置翻转的方法，具体操作如下：

选中模块，然后依次单击菜单“Format→Flip Name”，则标签位置将发生翻转。若原来位置在模块下方，则翻转到模块上方；若是左方则翻转到右方，如图 2-9 所示。



（a）改变标签位置前　　（b）改变标签位置后

图 2-9　改变模块标签位置

为了模型的简洁，对于一些功能显而易见的模块，其标签可以设置为不可见。具体操作如下。

选中模块，然后依次单击菜单“Format→Hide Name”即可，该模块的标签就在屏幕上消失了。注意，虽然标签消失了，但只是不可见，在模块的属性中，模块名并没有消失。

当模块名被隐藏后，菜单“Format”的原“Hide Name”的位置变为“Show Name”，选中它可以重新使标签变为可见。

（7）增加阴影

在建模过程中，若希望某个模块能够重点突出，引起其他用户的注意，可以对此模块增加阴影。实现方法如下：

选中模块，依次单击菜单“Format→Show Drop Shadow”，完成后的效果如图 2-10 所示。

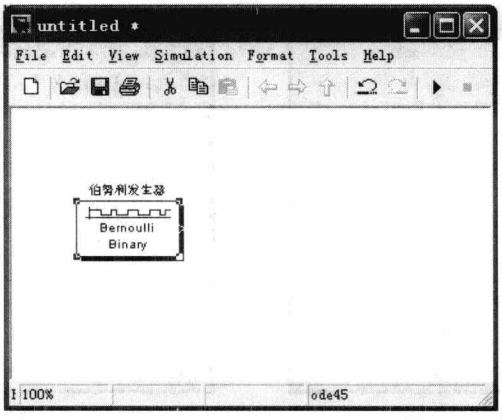


图 2-10　增加阴影后的模块

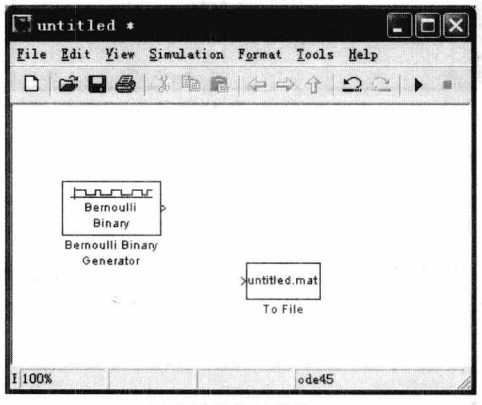
当模块被加上阴影后，菜单“Format”的原“Show Drop Shadow”的位置变为“Hide Drop Shadow”，选中它可以隐藏阴影。

2.2.3　信号线基本操作

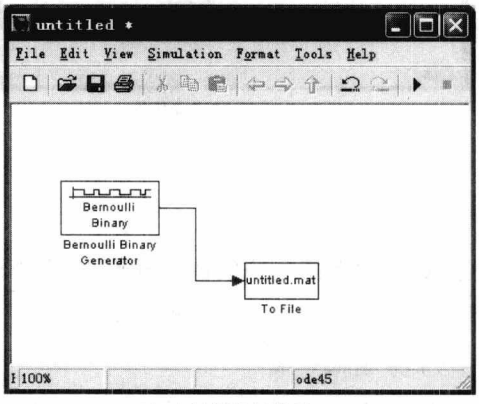
模型中有模块，还必须有信号线将模块联系起来才能够变成一个有机整体。模块和信号线是模型的骨架，模块和模型的参数设置是模型的灵魂。下面就对信号线的几个基本操作进行介绍。

（1）绘制信号线

把鼠标指向伯努利产生器右侧的输出端，当光标变为十字符时，按住鼠标左键，移向 To File 模块的输入端，放开鼠标左键就完成了两个模块之间的信号连线，如图 2-11 所示。



（a）绘制信号线前

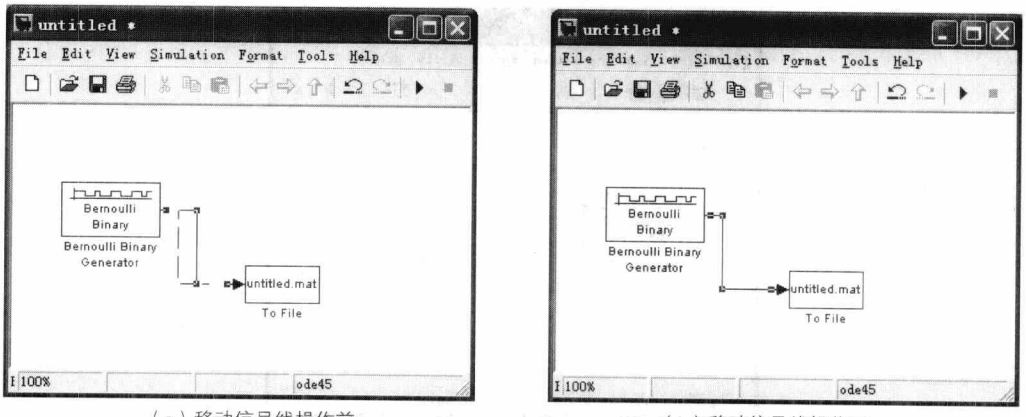


（b）绘制信号线后

图 2-11　绘制信号线

（2）移动信号线

若想移动信号线的某段，单击鼠标左键选中此段信号线，移动鼠标到这段信号线上，则此时鼠标的形状变为移动图标，按住鼠标左键，并且拖至新位置，放开鼠标左键，操作即完成。过程如图 2-12 所示。

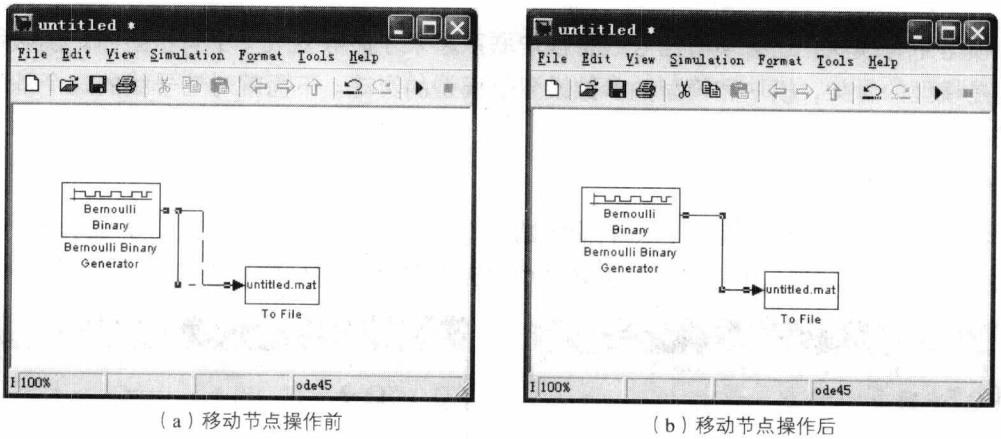


（a）移动信号线操作前　　（b）移动信号线操作后

图 2-12　移动信号线

（3）移动节点

要想移动节点，则需要先选中想移动的节点，选中后鼠标指针形状就会变为小圆圈，然后将节点拖至一个新位置，放开鼠标左键，操作即完成。过程如图 2-13 所示。



（a）移动节点操作前　　（b）移动节点操作后

图 2-13　移动节点操作

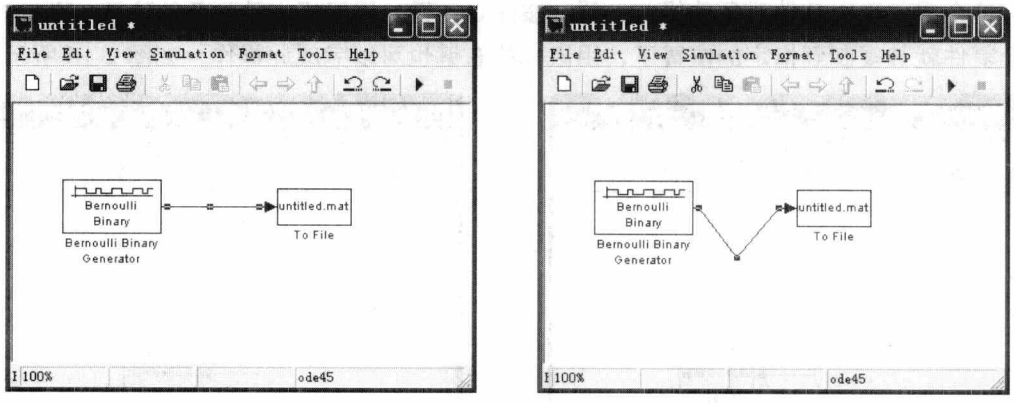
（4）删除信号线

其实，删除信号线的操作和删除模块一样，可以选中信号线后按 Delete 键，或者依次单击菜单“Edit→Clear”或者“Edit→Cut”进行删除。

（5）分割信号线

如其他操作一样，先选中信号线，然后按住 Shift 键，再在信号线需要分割的点上单击鼠标左键，鼠标指针的形状就会变为小圆圈，此时信号线会在此点上被分为两段，拖动

小圆圈至需要的位置，放开鼠标左键即可完成操作。过程如图 2-14 所示。



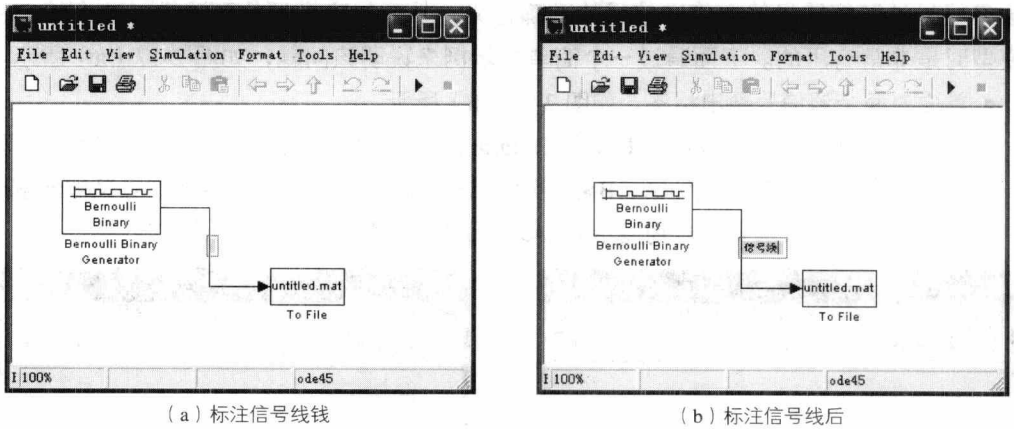
（a）分割信号线操作前　　（b）分割信号操作后

图 2-14　分割信号线

（6）信号线标签

每段信号线都可以有一个标签，以增强模型的可读性，标签可以放在信号线的结尾处或者两侧，同模块标签不同，信号线的标签并不是唯一的。

用鼠标左键双击要标注的信号线，则信号线的附近就会出现一个编辑框，如图 2-15（a）所示。在这个编辑框中输入标签的内容即可，如图 2-15（b）所示。



（a）标注信号线钱　　（b）标注信号线后

图 2-15　信号线标注

在标签外的任何位置上单击鼠标左键，则标签内容将被接受并在当前信号线的中间位置上显示。

（7）信号线标签的移动

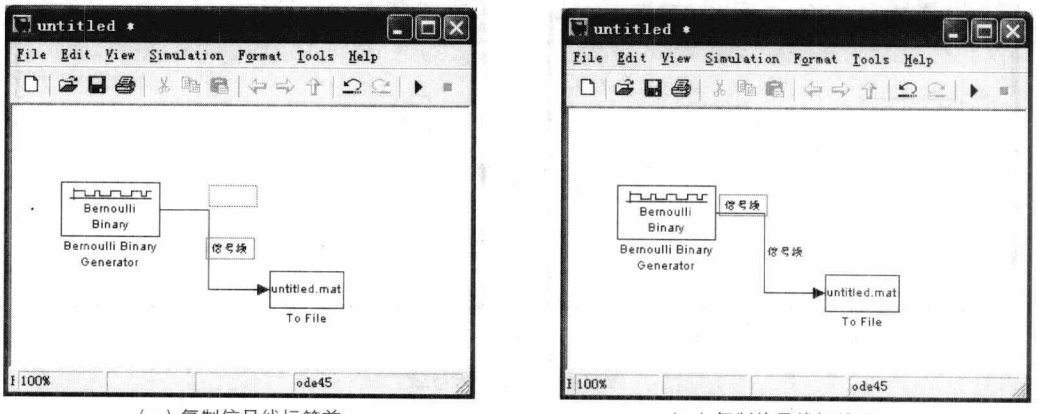
信号线标签如同其他对象一样，可以进行移动、复制等操作。信号线标签的默认位置是当前信号线线段中间，可以将它移动到此信号线附近的任何位置，具体操作如下：

- 移动信号线标签

选中要移动的信号线标签，拖动标签到需要的位置，最后放开鼠标左键即可。

- 复制信号线标签

这个操作也同其他对象的操作一样，按住 Ctrl 键，用鼠标拖动到新的位置，则标签的一个复件就被复制到指定的位置了，用鼠标的右键也可以完成此操作，如图 2-16 所示。



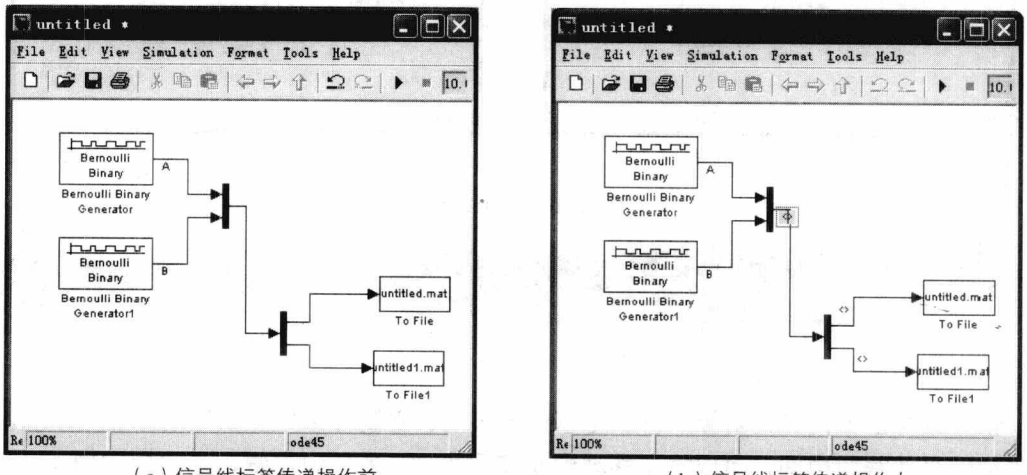
（a）复制信号线标签前　　（b）复制信号线标签后

图 2-16　复制信号线标签

（8）信号线标签的传递

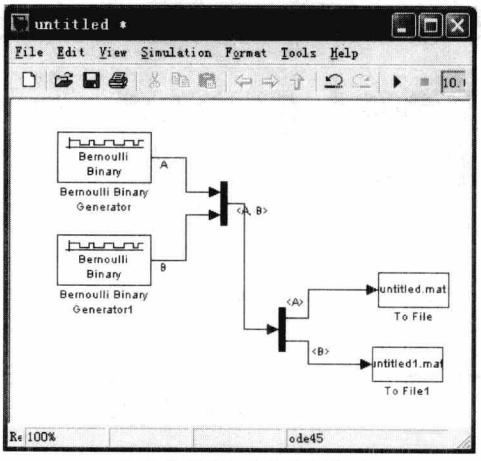
信号线标签具有一个特点，就是可以通过数个连接模块库中的模块对进行传递，例如，模块 Mux 与 Demux、Goto 与 From、Input 与 Output 等。通过信号线标签的传递，可以使读者轻松地了解信号线中传递信号的准确含义，从而提高模型的可读性。这个特点在编译模型时是非常有用的。下面通过一个简单的示例来说明该问题，如图 2-17（a）所示。

- 首先对两个源信号做出标签，如图 2-17（b）所示；
- 然后依次单击菜单中“Edit→Update Diagram”，则从 Mux 模块中输出的信号线包含两个信号的合成，故其标签为<A，B>。当信号通过 Demux 模块后，合成信号又被分解成两个标量信号，故信号线分别显示为<A>、<B>，如图 2-17（c）所示；



（a）信号线标签传递操作前　　（b）信号线标签传递操作中

图 2-17　信号线标签传递



（c）信号线标签传递操作后

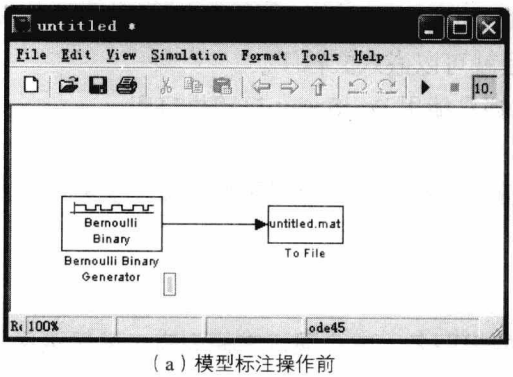
图 2-17　信号线标签传递（续）

2.2.4　模型的注释

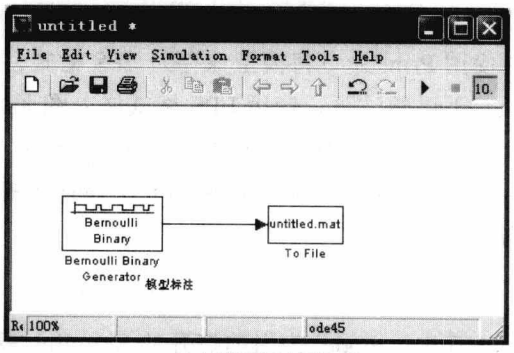
对所创建的模型文件，可以对其中的模型添加注释，这样可以提高模型的可读性，这种做法的效果如同 MATLAB 程序中的注释行一样，在此建议读者平时养成这种良好的习惯，今后将会受益匪浅。

（1）*增加注释的方法*

在模型窗口中的任何所需位置上双击鼠标左键就会出现一个编辑框。双击鼠标左键的位置将会成为此注释的中间部位，如图 2-18（a）所示。然后输入注释的内容，在窗口中的任何其他位置单击鼠标左键即可完成操作，如图 2-18（b）所示。当然，用户可以按照与模块操作相同的方法来对注释进行移动、复制等操作。



（a）模型标注操作前



（b）模型标注操作后

图 2-18　模型标注

（2）*改变注释字体*

有时需要对注释字体进行变换，则此时先选中注释，然后依次单击菜单 Format→Font，会出现一个字体选择对话框，如图 2-19 所示。然后用户可以选择认为合适的字体，单击“确定”按钮，则注释的字体就会发生相应的改变。

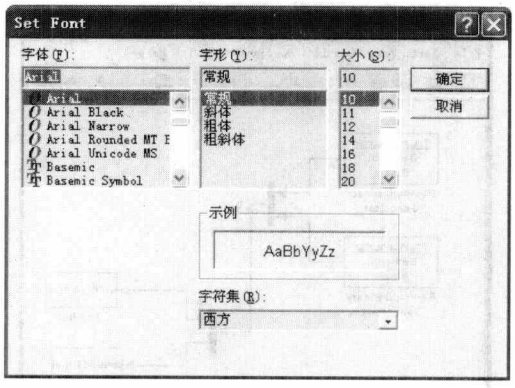


图 2-19　字体选择对话框

2.2.5　模型的打印

在建模完成以后，有时需要把建模好的模型打印出来，当然打印模型有许多不同的方法，最简单的就是将模型直接输出到打印机上。但是更常用的做法还是把模型粘贴到 Word 文件或其他文档中。

（1）菜单打印

打印模型最快速的方法就是将模型直接输出到打印机上。首先是选择模型窗口的“File→Print Setup”选项，并且设置打印机属性，然后选择“File→Print”就可以了。但是这种打印方法，Simulink 将按照页面的大小自动调整模型的大小，因此，用户没有办法自己设置模型打印的大小。

（2）嵌入文档中

Simulink 模型图可以以位图形式、Windows 图元文件形式或者 EPS 文件形式嵌入到文档中。如果是以位图形式、Windows 图元文件形式嵌入，则需要选择模型窗口的“Edit→Copy Model”命令，然后就可以嵌入文档中的适当位置；如果是以 EPS 格式嵌入的，则需要使用 Print 命令将模型保存为 EPS 格式，之后就可以嵌入文档了。

不过要注意，使用 EPS 格式嵌入图形的文档只能用 PostScript 打印机打印。

（3）使用 MATLAB 中的 print 命令

MATLAB 中的 print 命令可以将图形输出到打印机、剪贴板或各种格式的文档中。其命令格式如下：

```
print -smodel - device filename
```

其中，model 为当前打开的 Simulink 模型的模型名，它会显示在模型窗口的题名区。需要注意的是，当模型名中包含空格时，需要使用单引号将整个模型名引起来，如下面的命令：

```
print -s'Spring Mass System' -device filename
```

若模型窗口的题名区用两行来显示模型名，则在打印命令中用 ASC II 码 13 来连接，并且需要用方括号“[]”括上，例如下面的命令：

```
print -s['Damped'13 'Spring Mass System'] -device filename
```

其中，device 为输出设备的设备名，该设备包括打印机、文件和剪贴板等。表 2-1 列出了有效的设备类型。

表 2-1 有效的设备类型

Device 关键字	代表类型	Device 关键字	代表类型
ps	激光打印机	epsc2	2 级彩色 EPS 文件
psc	彩色激光打印机	win	当前打印机
ps2	2 级激光打印机	winc	当前彩色打印机
psc3	2 级彩色激光打印机	meta	图元格式的剪贴板
eps	EPS 文件	bitmap	位图格式的剪贴板
epsc	彩色 EPS 文件	setup	打印机设置
eps2	2 级 EPS 文件		

举个例子，在默认打印机中使用 MATLAB 的命令形式打印名为“ABC.mdl”的模型，并将其复制到剪贴板中，操作如下：

- 打开模型“ABC.mdl”，在 MATLAB 命令窗口中输入以下命令打印模型：

```
print -s 'ABC'
```

- 输入命令：

```
print -s 'ABC' -d 'meta'
```

将模型嵌入到 Word 或其他文档中。

2.2.6 模型文件

Simulink 中的各个模型都是可以适当修改的，不过这需要用户对 Simulink 有非常深入的了解。Simulink 的模型文件的后缀名为 mdl，其实可以显示为一组代码，主要由以下几部分组成：

- Model section：定义模型的参数；
- BlockDefaults sections：模型的默认设置；
- AnnotationDefaults sections：模型注解的默认值；
- System sections：顶层系统或者自系统的描述参数，包括 line、block 和 annotation 等。

2.3 子系统及其封装

对于简单的系统而言，可以直接建立系统的模型，并分析模块之间的相互关系以及模块的输入输出关系。当模型变得庞大和复杂时，就需要对模型进行归类、封装来简化它，也就是建立子系统（Subsystem）。使用子系统可以对模型提供 3 点好处：

- 能够减少模型窗口中显示的模块数；

- 可以把实现某一功能的所有模块封装到一起，形成一个整体的模块；
- 建立起一个分层次的清晰的模块结构，比如子系统是一层，那么构成这个子系统的模块就是下一层，利于对整个模型的管理和更新。

本节首先利用两个示例来说明建立一个最简单子系统的方法，其次介绍条件子系统的建立与使用，然后讲解使用 mask editor 对子系统进行封装，最后介绍建立用户自己的模块库的方法。

2.3.1 创建简单子系统

构建一个子系统有两种方法：一是从基本模块库中复制一个空的子系统模块，再把需要封装在一起的模块加入到空的子系统模块中；二是直接对现有的模块进行子系统封装。下面分别对这两种方法进行介绍。

（1）直接生成子系统

直接生成子系统就是对已存在于模型的部分或全部模块进行压缩，形成一个整体的子系统。

首先，要将封装成子系统的模块调整好位置，以利于框选。再按住鼠标左键拖出选择框，选中需要封装的部分，包括模块和信号线，如图 2-20 所示。

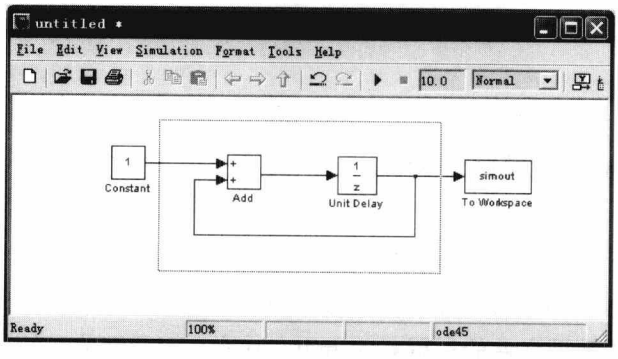


图 2-20　框选要封装的模块及信号线

然后，依次单击菜单“Edit→Create Subsystem”或单击鼠标右键选择弹出菜单“Create Subsystem”，便得到了一个子系统，调整它的大小和位置，如图 2-21 所示。

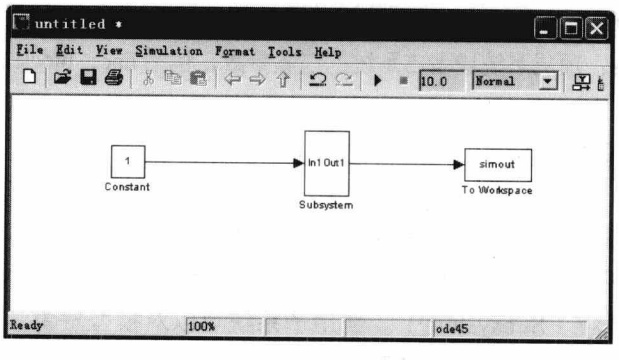


图 2-21　完成后的模型

这样，一个简单的子系统就建立起来了。想查看或编辑子系统内部的模块，只需要用鼠标左键双击该子系统即可，如图 2-22 所示。

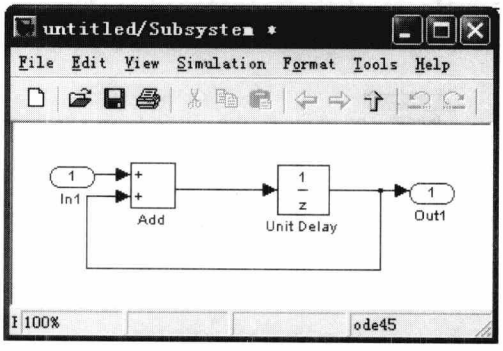


图 2-22　被封装的子系统

对于子系统的操作和其他模块的操作完全一样。如果觉得它的外观并不能表现出子系统的特征或功能，或者想让子系统和其他模块一样有自己的参数设置，别着急，在后面的章节中我们将会介绍对子系统的封装技术，完全可以满足用户对它的要求。

（2）使用 Subsystem 模块建立子系统

建立子系统就像编写子函数一样，先写好实现特定功能的子函数以供调用。Simulink 用户也可以使用 Subsystem 模块编辑某一特定用途的子系统，这需要在建立模型前对整个模型有一个很好的规划。

下面以建立一个随机复数发生器为例介绍这种方法。

首先，从“Simulink→Ports & Substem”模块库中复制 Subsystem 模块模型，如图 2-23 所示。

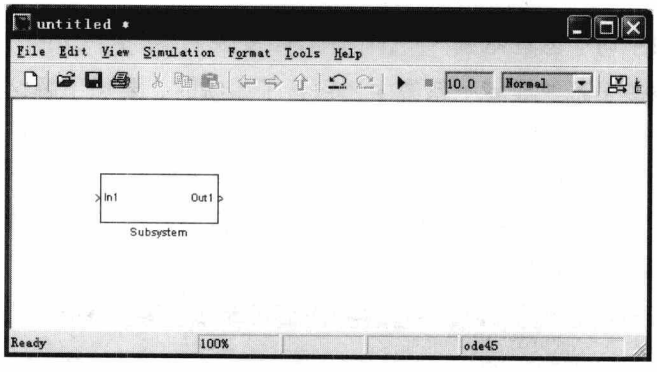


图 2-23　子系统模块

双击打开子系统模块，就可以对子系统的内部进行编辑了。编辑子系统，并复制如图 2-23 所示的模块到子系统中，如图 2-24 所示。

最后的模型如图 2-25 所示。

仿真后，simout 的值为一列随机复数。

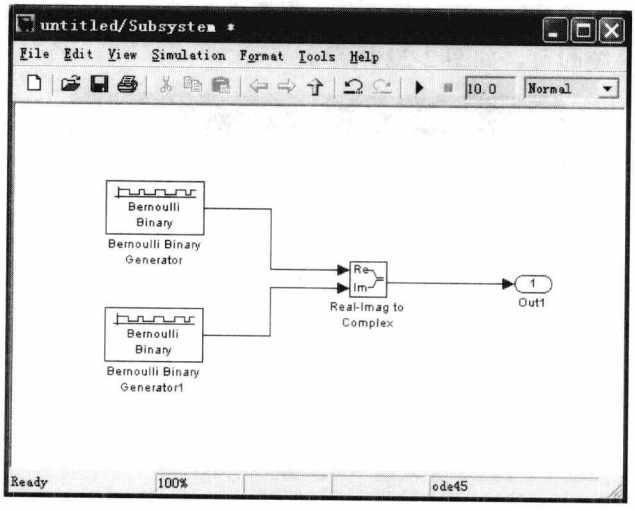


图 2-24　编辑后的子系统内部

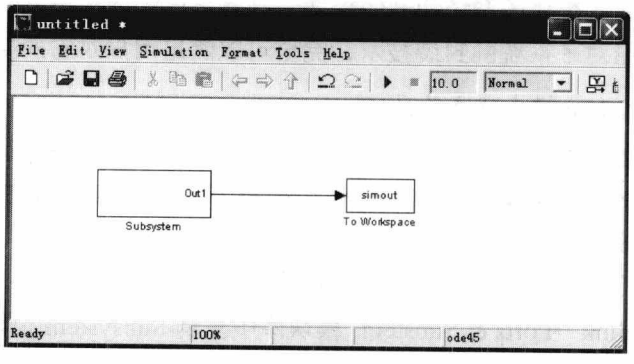


图 2-25　完成后的模型

（3）查看模型中的子系统结构

当模型中有很多子系统或者子系统中还包含子系统时，要想了解这些子系统的结构和主从关系可以有一个很直观的方法，就是使用“Model Browser”。依次单击菜单“View->Model Browser Options->Model Browser”，或单击模型窗口中的 图标，就会显示如图 2-26 所示的“Model Browser”窗口。这类似于 Windows 的资源管理器，可以很方便地查看子系统的构成及其所包含的模块。

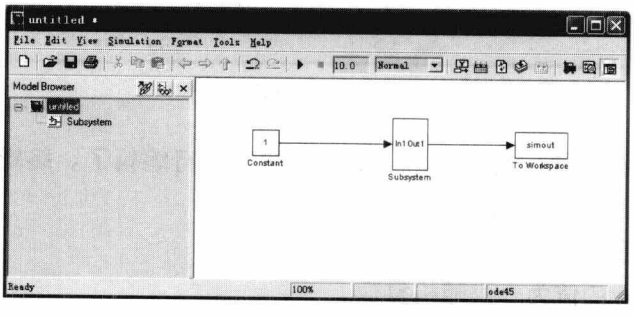


图 2-26　Model Browser 窗口

2.3.2　创建条件执行子系统

条件执行子系统，顾名思义就是依靠一个输入信号作为判断条件来控制执行与否的子系统。这个输入信号称为控制信号，它从条件执行子系统的一个给定的端口即控制端口输入。当控制信号满足特定的条件时，子系统开始执行。

根据控制信号对条件子系统控制方式的不同，可以将条件执行子系统划分为以下几种基本类型：

- 使能子系统

当控制信号为正时，子系统处于执行状态。

- 触发子系统

触发子系统在每个触发事件发生时执行一次。控制信号在这里也可以被称作触发信号，触发信号可以是连续的或者是离散的。根据设定，触发子系统可以在触发信号的上升沿、下降沿或者双边沿时被触发执行。

- 触发使能子系统

触发使能子系统在使能信号为正且触发事件发生时，子系统开始执行。

- 控制流程子系统

这种子系统类似于编程语言中的 for 语句、if 语句、while 语句等。用户可以创建相应的子系统来达到类似上述语句的功能。

（1）使能子系统

当控制信号为正时使能子系统处于执行状态。在控制信号从负值经过零点变为正值时，子系统开始执行，直到控制信号重新回到零或负值。

可以在一个普通子系统中加入一个使能端模块（Ports & Subsystems→Enable）来获得一个使能子系统，或者直接从模块库中复制一个空的使能子系统到模型中再对它进行编辑（Ports & Subsystems→Enable Subsystem）。

下面给出一个使能子系统的简单示例，以便有助于读者的理解。

【示例 2.1】 使能子系统。

构建如图 2-27 所示的使能子系统，子系统内部如图 2-28 所示。

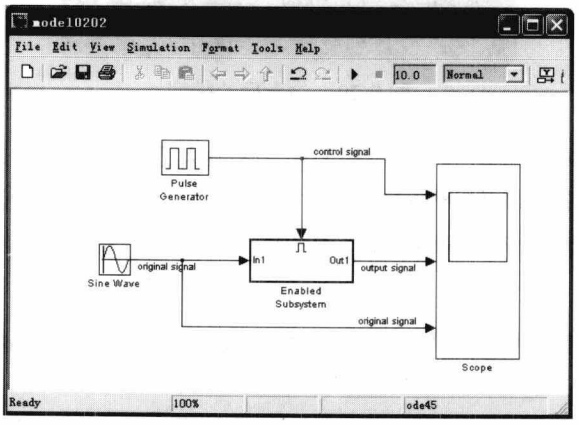
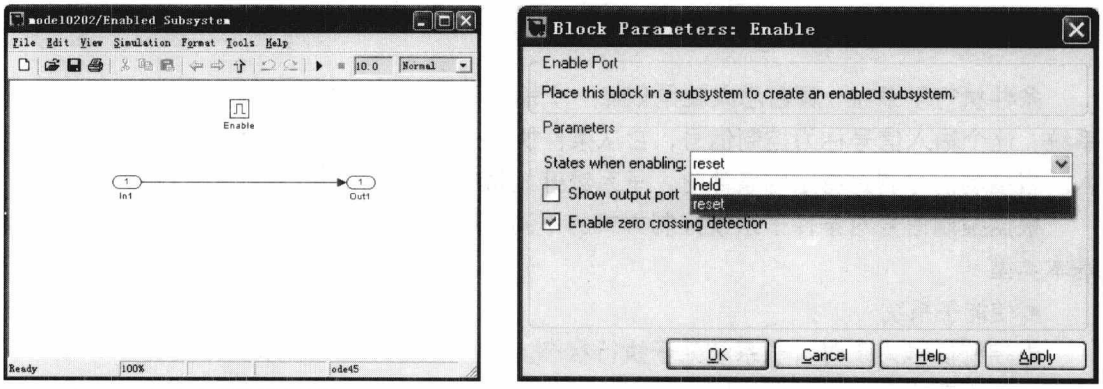


图 2-27　构建的模块



（a）使能子系统内部　　（b）使能模块的参数窗口

图 2-28　使能子系统内部及使能模块的参数窗口

各模块的设置如下：

- 控制信号为脉冲发生器，会周期性地产生持续一定时间的脉冲信号；
- 输入信号为正弦波；
- 子系统的功能为使正弦波的输出向上搬移 2；
- 示波器 Scope 模块设置坐标轴为 3，使其能够同时显示 3 组信号。

从仿真结束后的波形图（如图 2-29 所示）中可以看出，当使能信号为正时，输出值随着时间变化；而当使能信号为负时，输出信号维持不变。

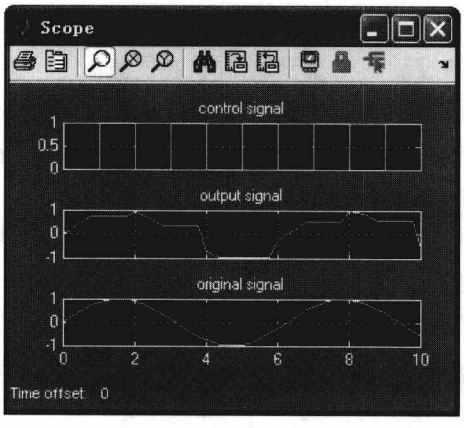


图 2-29　仿真后的模型

在如图 2-28 所示的使能信号属性窗口中，使能状态可以被设置为“reset”或“held”。选择“reset”表示在使能子系统开始执行时，系统的状态保持不变。选择“Show output port”选项，则使能模块会出现一个输出端口，供用户监视使能信号的状态。

（2）触发子系统

触发子系统在触发信号的符号发生改变即出现过零现象时开始执行。触发子系统有以下三种触发类型：

- 上升沿触发子系统：系统在触发信号出现上升沿时开始执行；

- 下降沿触发子系统：系统在触发信号出现下降沿时开始执行；
- 双边沿触发子系统：系统在触发信号出现上升沿或下降沿时都开始执行。

同建立使能子系统的方法一样，触发子系统也能够通过复制一个触发端口到普通子系统中来获得触发子系统（Ports & Subsystems→Trigger），或是对一个空的触发子系统内部进行相应的编辑（Ports & Subsystems→Trigger Subsystem）。

下面给出一个包含 3 种触发类型的示例，供读者理解。

【示例 2.2】构建如图 2-30 所示的仿真模型。子系统内部如图 2-31 所示。3 个触发子系统中没有加入模块，只是完成对输入信号的条件输出。

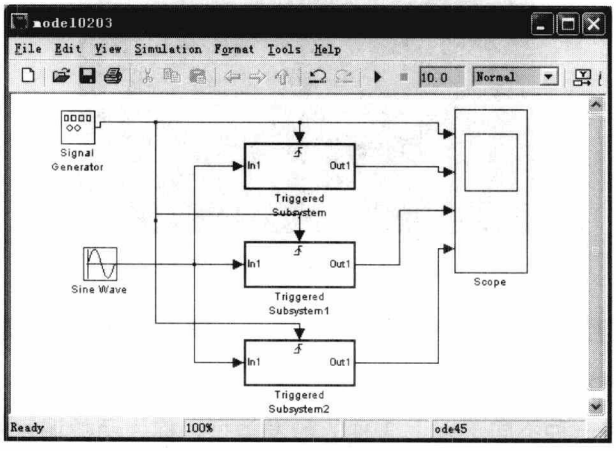
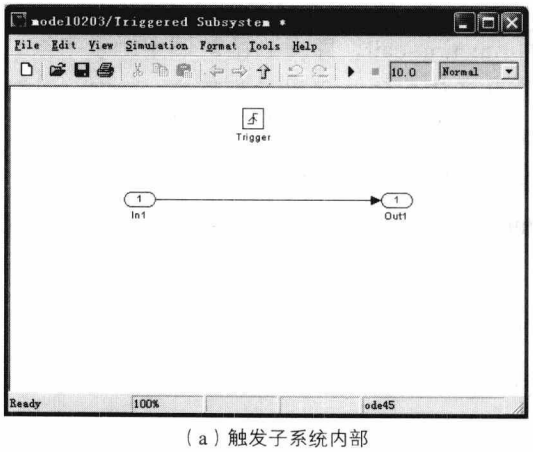
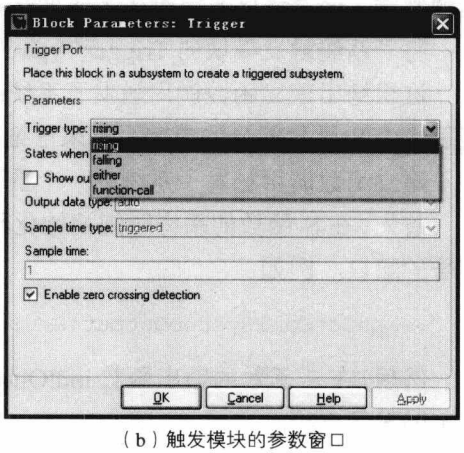


图 2-30　包含 3 种类型触发子系统的模型



（a）触发子系统内部



（b）触发模块的参数窗口

图 2-31　触发子系统内部及使能模块的参数窗口

各模块设置如下：

- 触发信号为方波发生器，周期为 1，产生值为-1 和 1 的方波信号；
- 输入信号为正弦波；
- 3 个子系统中触发模块的 Trigger Type 分别设置为 rising、fading 和 either，分别代表 3 种类型触发类型；

- 示波器 Scope 模块设置坐标轴数为 4，使其能够同时显示 4 组信号。

值得注意的是，触发子系统的输出具有零阶保持特性，即输出结果不变。当系统被触发时，输出当前的值，并持续输出这个值直到下一个触发时刻的到来。最后的输出波形如图 2-32 所示。

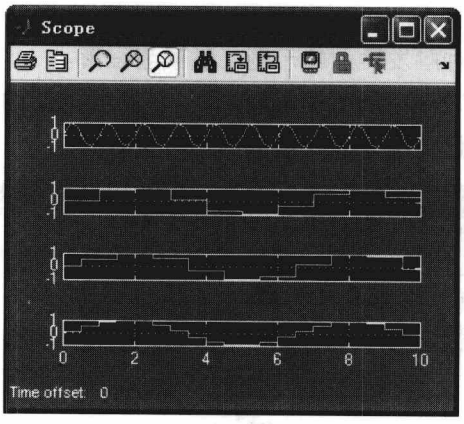


图 2-32　仿真后的波形图

从波形上看，3 种类型的触发子系统对于同一个输入信号的输出是不同的。

（3）函数调用触发子系统

触发子系统还有一种不同于以上 3 种的触发类型，称为函数调用触发子系统，即通过函数调用的方式来触发子系统执行。用户可以通过编写 S-函数，并引出输出端口到子系统的触发端，通过程序来控制子系统的运行。

将子系统触发模块的 Trigger Type 设置为 function-call，即得到函数调用触发子系统。

这里给出建立函数调用触发子系统的示例和所使用的 S-函数的源程序，读者可以在阅读了本书 S-函数部分的内容后再来仔细阅读本节内容。

建立函数调用触发子系统的一个重要步骤就是编写函数调用的 S-函数。

首先，在 S-函数的采样时间初始化函数 mdlInitializeSampleTimes 中指定作为输出触发信号的端口。例如：

```
>> ssSetCallSystemOutput(S,0); %%%设置第一个输出端口为触发端口
```

然后，在 S-函数的输出函数 mdlOutputs 或状态更新函数 mdlUpdate 中使用下面这条语句来触发子系统。

```
>> ssSetCallSystemWithTid(S,0,tid); %%%调用第一个端口来触发子系统
```

【示例 2.3】建立如图 2-33 所示的模型，Scope 设置为 3 个输入端口，触发子系统的触发模块设置为 function-call，其他参数保持默认值。

S-函数的功能是当输入 S-函数模块的数据为 1 时，通过语句触发子系统执行。仿真后的示波器波形如图 2-34 所示。读者可以自行分析所得到的结果。

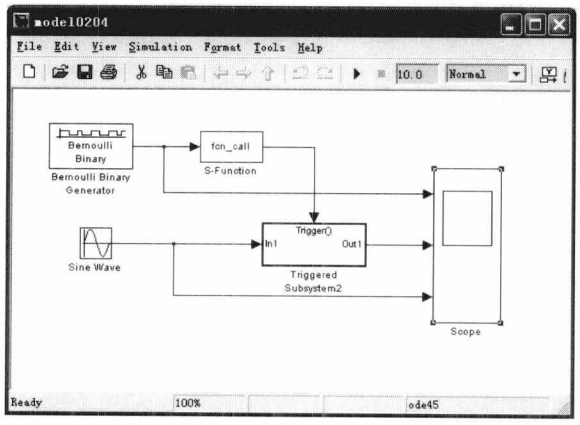


图 2-33　函数调用触发子系统

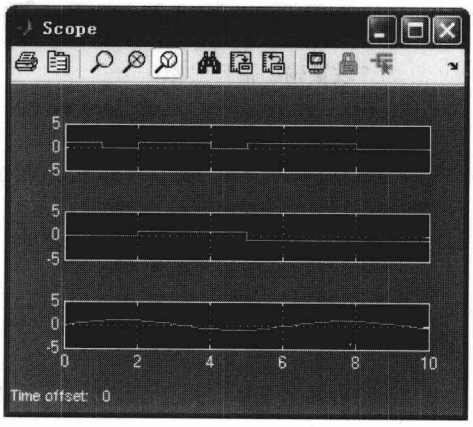


图 2-34　函数调用触发子系统仿真后的波形图

从图中可以看到，在第 0 秒时 S-函数模块输入为 1，此时子系统输出 0 秒时输入的信号并保持。在第 1 秒时 S-函数模块输入为 0，则子系统不改变输入信号，仍然保持 0 秒时的输出。在第 2 秒时 S-函数输入为 1，则子系统输出第 2 秒时的输入信号并保持。依次类推，读者可以自行分析。

S-函数的源程序如下所示。

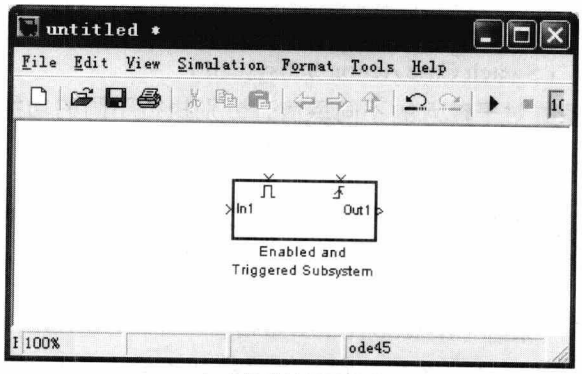
```
#include S_FUNCTION)NAME fcn_call
#include S_FUNCTION_LEVEL 2

#include "simstruc.h"
%Function:mdlInitializeSizes ===================%
static void mdlInitializeSizes(SimStruct *S)
{
    %设置额外参数的个数
    ssSetNumSFcnParams(S,0);
    %如果不一致则返回
    if(ssGetNumSFcnParams(S)!=ssGetSFcnParamsConunt(S))
    {
        return;
    }

    %设置连续状态个数
    ssSetNumContStates(S,0);
    %设置离散状态个数
    ssSetNumDiscStates(S,0);

    %设置输入端口个数
    if(!ssSetNumInputPorts(S,1)) return;
    %设置输入端口宽度
    ssSetInputPortWidth(S,0,1);
    %设置输入端口的元素是否存放在连续的内存中
    ssSetInputPortRequiredContiguous(S,0,true);
    %设置直接反馈值 flag(1=yes,0=no).
```

```
    ssSetInputPortDirectFeedThrough(S,0,1);

    %设置输出端口个数
    if(!ssSetNumOutputPorts(S,1)) return;
    %设置输出端口宽度
    ssSetOutputPortWidth(S,0,1);

    ssSetNumSampleTimes(S,1);
    ssSetNumRWork(S,0);
    ssSetNumIWork(S,0);
    ssSetNumPWork(S,0);
    ssSetNumModes(S,0);
    ssSetNumNonsampledZCs(S,0);

    ssSetOptions(S,0);
}
% Function:mdlInitializeSampleTimes ==================%
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSample(S,0,INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S,0,0,0);
    %设置第一个输出端口为触发端口
    ssSetCallSystemOutput(S,0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{

}
#endif /* MDL_INITIALIZE_CONDITIONS */

% Function:mdlOutputs ===================%
static void mdlOutputs(SimStruct *S,int_T tid)
{
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T;
    *y = ssSetOutputPortSignal(S,0);
    %调用第一个端口来触发子系统
    if(u[0] == 1)
        ssCallSystemWithTid(S,0,tid);
}

% Function:mdlTerminate ===================%
static void mdlTerminate(SimStruct *S)
{

}

%如果此函数编译成 MEX 文件则连接 Simulink.c 文件
```

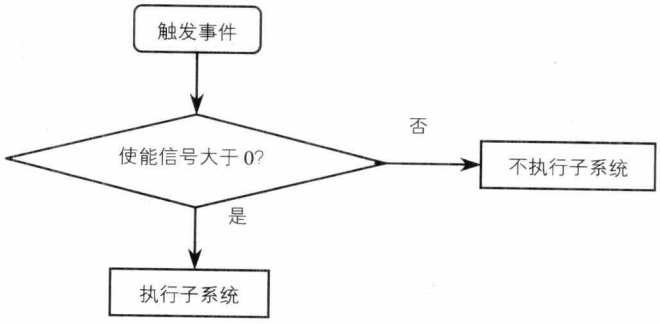
```
#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
%否则,连接 cg_sfun.h 文件
#include "cg_sfun.h"
#endif
```

（4）触发使能子系统

对于某些条件执行子系统而言，其控制信号可能不止一个。在很多情况下，条件执行子系统同时具有触发控制信号与使能控制信号，这样的条件执行子系统一般称之为触发使能子系统。顾名思义，触发使能子系统指的是子系统的执行受到触发信号与使能信号的共同控制，也就是说，只有当触发条件与使能条件均满足的情况下，子系统才开始执行。如图 2-35 所示为触发使能子系统的模块与工作流程。



（a）触发使能子系统



（b）触发使能子系统工作流程

图 2-35　触发使能子系统的模块与工作流程

在仿真时，系统处于等待状态，当触发事件发生后，Simulink 会检测使能信号，如果使能信号为正，则子系统执行一次，否则不执行。由此可知，只有当触发条件与使能条件同时满足时，子系统才能够被执行。

对于使用多个控制信号的条件子系统，用户一定要记住：条件执行子系统中不允许同时出现多个触发或者使能信号。如果必须使用多个控制信号，用户可以使用逻辑操作符，先将相关的控制信号组合，以产生单一的触发控制信号或使能控制信号。

（5）其他条件执行子系统

除了以上介绍的常用条件子系统外，Simulink 的 Subsystem 模块库中还包含一些其他类型的条件子系统，以供用户在特定的条件下使用。这里对这些模块分别作简单的介绍。

- 可配置子系统（Configurable Subsystem）

用来代表用户自定义库中的任意模块，只能在用户自定义库中使用。

- for 循环子系统（For Iterator Subsystem）

for 循环子系统的目的是在一个仿真时间步长之内循环执行子系统。用户指定在一个仿真时间步长内子系统执行的次数，以达到某种特殊的目的。

- while 循环子系统（While Iterator Subsystem）

与 for 循环子系统类似，while 循环子系统同样可以在一个仿真时间步长内循环执行子系统，但是其执行必须满足一定的条件。while 循环子系统有两种类型：当型和直到型，这与其他高级语言中的 while 循环类似。

- 选择执行子系统（Switch Case Action Subsystem）

在某些情况下，系统对于输入的不同取值，分别执行不同的功能。所谓的选择执行子系统，与 C 语言中 switch...case 语句的功能类似。

选择执行子系统必须同时使用 switch...case 模块与 Switch Case Action Subsystem 模块。

- 表达式条件执行子系统（If Action Subsystem）

为了与前面的条件执行子系统相区别，这里称 If Action Subsystem 为表达式执行子系统。此子系统的执行依赖于逻辑表达式的取值，这与 C 语言中的 if...else 语句类似。

表达式执行子系统必须同时使用 if 模块与 If Action Subsystem 模块。

2.3.3 子系统的封装

前面介绍了简单和条件子系统的构建方法。但是可以看出这两种子系统的外观基本相同，不能体现该子系统的特点及功能。下面的章节就使用子系统的封装技术，让一个子系统具有自己的特点。封装后的子系统可以有自己的图标、自己的参数和带有功能描述的控制对话框，甚至可以有自己的帮助文档，使其看起来与真正的模块毫无区别。而当用户双击该模块时，将出现一个描述及控制对话框，能够很好地屏蔽子系统的内部结构，保护内部结构不会被轻易修改。

（1）一个子系统封装的示例

【示例 2.4】下面将给出一个子系统封装的示例，让读者对如何封装子系统有一个具体的印象，并帮助读者理解封装子系统的详细内容。

首先，按照之前介绍的方法生成一个子系统。这里用如图 2-36 所示的已经生成好的一个子系统对其进行封装。这是一个生成随机复数的子系统，对它封装之后，使其成为一个具有自己的图标、对话框和文档的 Simulink 模块。

将框内组件编辑为子系统并调整各组件的位置，如图 2-37 所示。

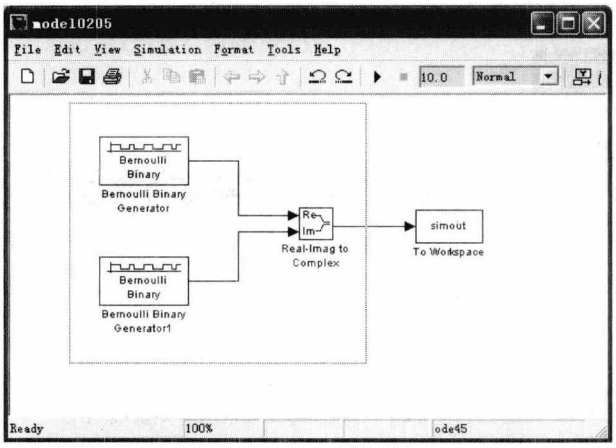


图 2-36　子系统示例

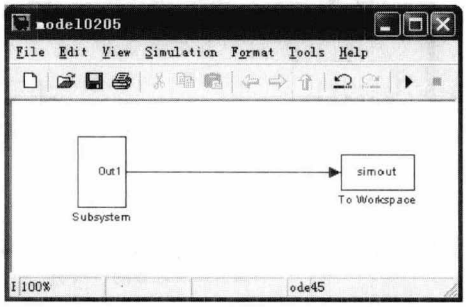


图 2-37　编辑成子系统后模型

封装子系统的基本过程如下：

- 选中要封装的子系统，用鼠标右键单击该模块，在弹出的菜单中选择“Mask Subsystem”选项，将弹出如图 2-38 所示的对话框（Mask Editor）。关于该对话框的说明将在下面章节中详细介绍。

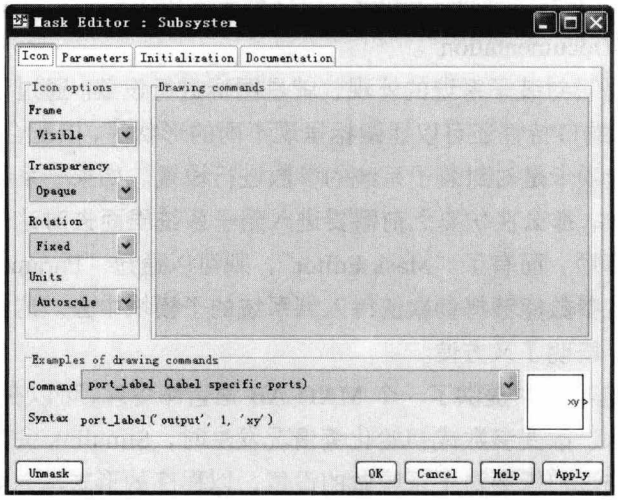


图 2-38　Mask Editor 对话框

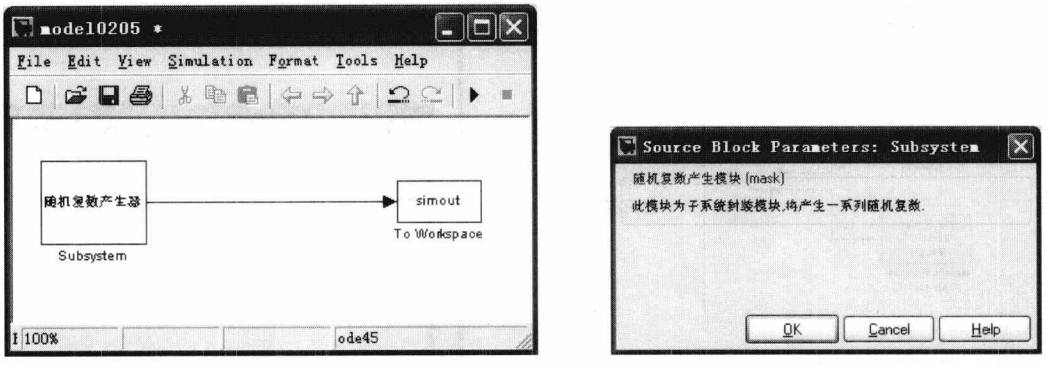
- 在 Drawing commands 栏中写入代码。

```
disp('随机复数产生器')
```

这条代码的作用是：在封装后的子系统的图标中央显示“随机复数产生器”几个文字。

- 选择 Documentation 选项卡，在“Mask type”栏中写入该模块的描述名称“随机复数产生模块”。在“Mask description”栏中写入对模块的描述“此模块为子系统封装模块，将产生一列随机复数”。

- 最后单击“OK”按钮，这样一个子系统就封装完成了，封装之后的结果如图 2-39 所示。



（a）封装之后的子系统　　（b）双击子系统后出现的模块描述对话框

图 2-39　子系统的封装

（2）Mask Editor 介绍

Simulink 提供了一个可以实现设置一些常用 Mask 属性的对话框，就是示例 2.4 中所使用的“Mask Editor”对话框。通过对“Mask Editor”进行相应的设置，就能封装出满足基本需要的子系统模块。

从图 2-38 中可以看到，“Mask Editor”对话框有 4 个选项卡：“Icon”、“Parameters”、“Initialization”和“Documentation”。

“Icon”选项卡是对封装子系统的外观也就是图标进行设置。通过设置可以在图标中显示文字或图形，标示端口特性还可以让图标呈现不同的形状等，以配合子系统的功能表现。

“Parameters”选项卡是对封装子系统的参数进行设置。如果子系统中有一个或多个模块需要手动设置参数，那么在仿真之前需要进入到子系统里面去对这些模块分别进行参数设置，这样显然很麻烦。而有了“Mask Editor”，则可以通过“Parameters”选项卡对子系统设置参数，使这些参数能够将参数值传入到系统的子模块中去。这样，就可以直接对子系统进行参数设置，既明了又方便。

“Initialization”选项卡中提供了一个 MATLAB 语言命令库，可以在其中写入一些程序，当该子系统有被载入、改变参数或初始化等情况发生时，Simulink 会自动执行这些程序。所以，可以将一些仿真前需要对子系统做的设置，以程序的形式写入到编辑框内来实现。

“Documentation”选项卡可以编辑对子系统的描述和生成帮助文档，帮助使用者更好地了解及使用该子系统。描述会显示在模块参数对话框中，帮助文档则嵌入到 MATLAB 的内置帮助中。

下面，将分别举例说明“Mask Editor”中各个选项卡的具体用法。

（3）编辑子系统图标

使用“Mask Editor”对子系统的图标进行编辑，可以让封装后的子系统有能够表现自身功能特点的图标，使封装后的子系统更加专业。这仅仅需要在“Icon”选项卡中进行相应的设置和编程即可达到目的。

前面给出图 2-38 就是“Icon”选项卡，从中可以看出，该选项卡主要由两部分组成，即图标绘制命令栏（Drawing Commands）和图标选项栏（Icon Option）。

- 图标绘制选项栏（Drawing Commands）

封装后，子系统模块的图标均是在图标绘制命令栏中绘制完成的。使用不同的 MATLAB 命令可以生成不同的图标。如文本描述、系统状态方程、图形显示和显示图像等。当然，如果在该栏中输入多条绘制命令，则图标会按照命令的顺序来显示。

① 在图标中显示文本描述

使用如下的命令可以实现在图标中显示文本的功能：

```
disp('text')       %%%在模块中央显示单引号中的文本
disp(str)          %%%在模块中央显示字符串变量 str 的值
text(x,y,'text')    %%%在由坐标（x,y）指定的位置显示文本，坐标原点为图标的左下
fprintf('format','text')   %%%format 表示文本格式
```

另外，还可以对封装后的子系统的端口进行文本标示。利用下面的命令来实现：

```
port_label(port_type,port_number,label);
```

其中，第一个参数 port_type 用来指定端口类型（输入或输出端口），取值为 input 或 output；第二个参数 port_number 用来指定端口号，端口号从 1 开始计起；第三个参数 label 为要显示的端口描述文本。

如下示例所示：

```
disp('模块描述')
port_label('input',1,'In');
port_label('output',1,'Out');
```

显示结果如图 2-40 所示。

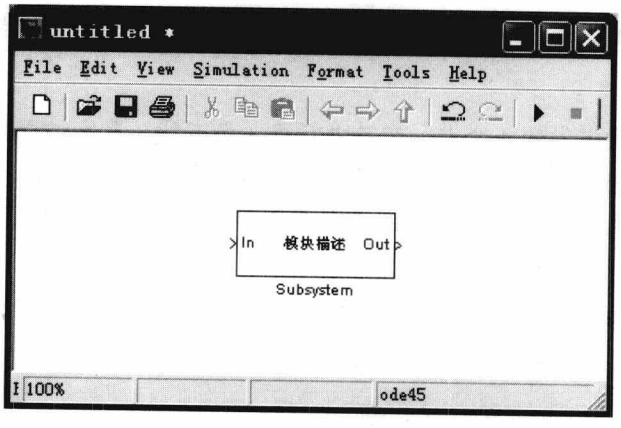


图 2-40　图标中的文本描述

② 在图标中显示系统状态方程

使用 dpoly 或 droods 命令，能够在封装子系统的图标中显示系统的状态方程或系统零极点传递函数。其命令格式如下：

```
dpoly(num,den,'variable')
droots(z,p,k)
```

其中，num 和 den 分别为状态方程的分子和分母多项式；variable 为系统状态变量；z、p、k 分别为零点、极点和系统增益。

请看下面的示例，在图标绘制命令栏中写入如下命令：

```
>> dpoly([1],[1 2 1],'s')
```

得到的结果如图 2-41 所示。

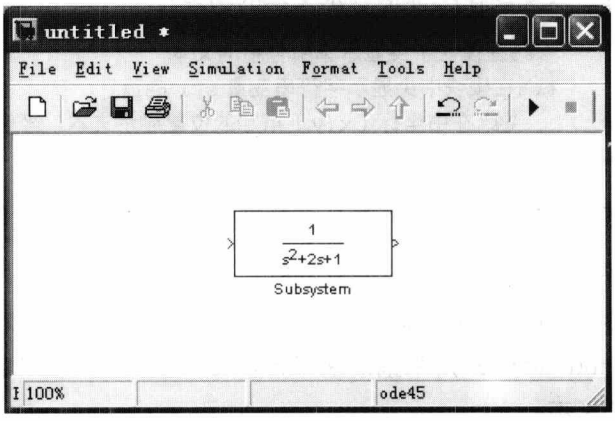


图 2-41　图标中显示系统状态方程

③ 在图标中显示图形

可以在图标中显示两种图形，分别是自绘图形和图片图形。

在图标中显示自绘图形。通过绘图函数 plot 在图标中绘制图形。plot 函数根据参数提供的坐标点绘制出相应的图形，下面给出一个在图标中显示自绘图形的例子。

本例将在图标中绘制出一个三角形，具体实现为在图标绘制栏中写入下面这条语句：

```
>> plot([0 0 5 0],[0 10 5 0]);
```

函数会根据参数提供的 4 个点，在其中间画 3 条直线。最终结果如图 2-42（a）所示。

在图标中显示图片图形。在图标中显示图片可以通过函数 image 和 imread 来实现。imread 函数的功能为将某一 bmp 格式的图像读入到一个三维矩阵中，image 函数则将三维矩阵在图标中画出。下面给出一个示例。

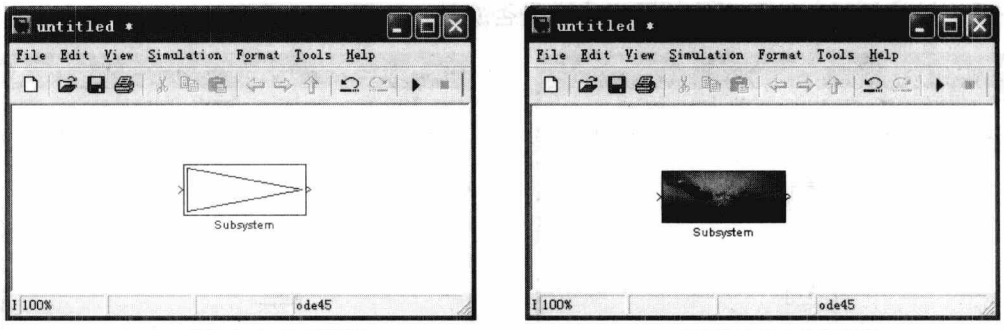
本例将名为“Sunset.bmp”的图片显示在图标中，在图标绘制栏中写入下面这条语句：

```
>> image(imread('Sunset.bmp'));
```

得到如图 2-42（b）所示的结果图像。

- 图标选项栏（Icon option）

以上介绍了图标绘制命令栏，通过在其中写入相应的语句，能够实现图标的不同显示内容。下面来介绍图标选项栏，通过对选项栏的选项进行设置，可以使图标的显示更加多样化。

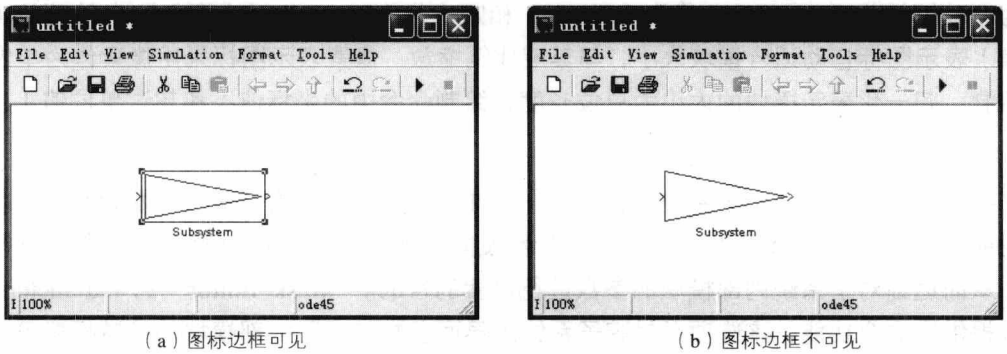


（a）在图标中显示自绘图形　　（b）在图标中显示图片图形

图 2-42　在图标中显示图形

① 图标边框设置（Frame）

该选项可以设置图标的边框是否可见，Visible 为可见（默认值）；Invisible 为不可见。如图 2-43 所示，（a）为可见，（b）为不可见。

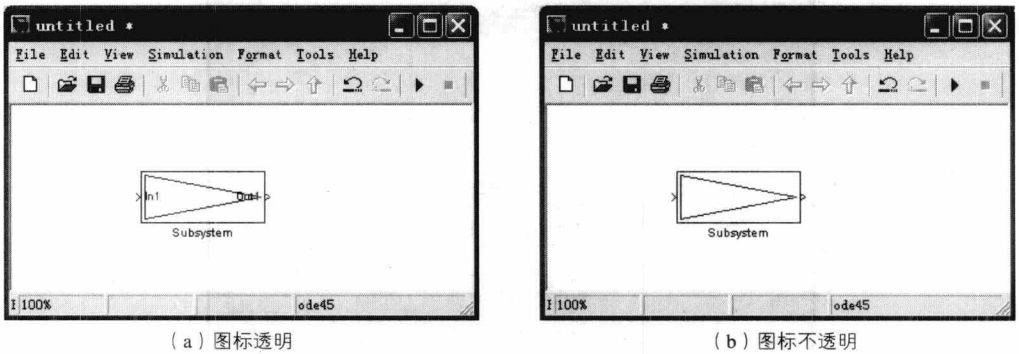


（a）图标边框可见　　（b）图标边框不可见

图 2-43　设置图标边框是否可见

② 图标透明设置（Transparency）

该选项设置图标是否透明显示，Transparency 为透明（默认值）；Opaque 为不透明。图标透明可以显示图标后面的文字、端口标签等信息，如图 2-44 所示。



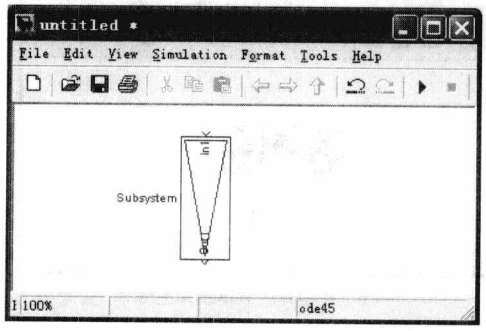
（a）图标透明　　（b）图标不透明

图 2-44　图标透明设置

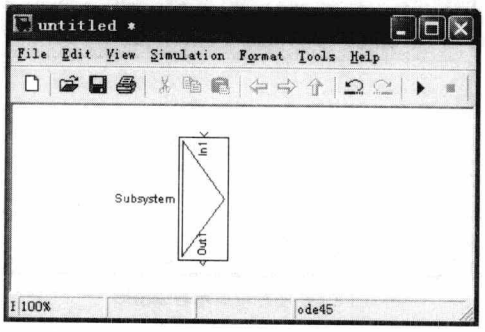
③ 图标旋转性设置（Rotation）

该选项设置图标内容是否可旋转，Fixed 为不可旋转（默认值）；Rotates 为可旋转。即

当子系统被旋转时，该子系统的图标内容是否旋转，如图 2-45 所示。



（a）可旋转



（b）不可旋转

图 2-45　图标旋转性设置

④ 图标绘制坐标系单位设置（Units）

该选项设置图标绘制命令所使用的坐标系单位，仅对 plot 与 text 命令有效。其选项分别为自动缩放（Autoscale）、像素（Pixels）和归一化（Normalized）表示。自动缩放（默认值）表示图标自动适合模块大小，与其成比例缩放；像素表示图标采用像素作为单位进行绘制；归一化表示模块的大小为单位长度，绘制命令中的坐标值不能超过单位值 1。

（4）为子系统增加参数

在通常情况下，一个子系统中会包含一个或多个需要在仿真前进行参数设置的模块。而封装后的子系统能够提供一个友好的参数设置界面，可以通过对界面中的参数进行设置，从而达到对子系统内部模块的参数进行设置的目的。“Mask Editor”的 Parameters 选项卡提供了这个功能，该选项卡封装子系统的界面参数与实际内部模块的参数相联系，从而屏蔽了系统内部参数，提供了单一的、直观的参数设置接口。

如图 2-46 所示的是“Mask Editor”的 Parameters 选项卡。

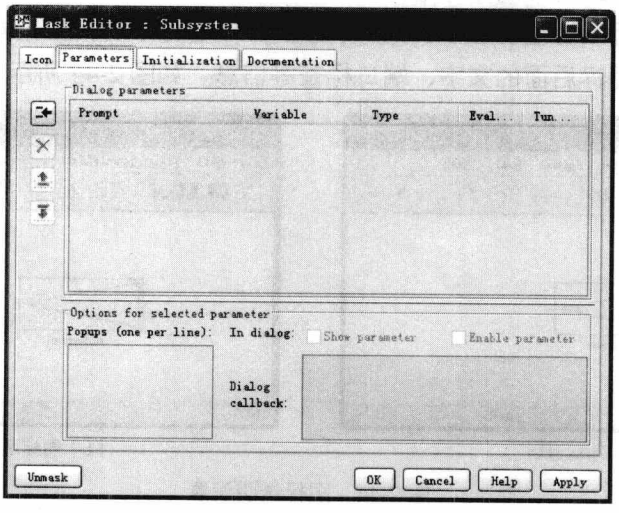


图 2-46　Mask Editor 的 Parameters 选项卡

该选项卡中有两个设置栏，分别是参数对话框（Dialog parameters）和参数选项栏

（Options for selected parameter）。在参数对话框的左边有 4 个按钮，由上至下分别是增加参数、删除参数、上移和下移。

在参数设置对话框中每增加一个参数，子系统的控制对话框上就会多出一个参数输入框，对每一个参数进行设置时，各参数的含义如表 2-2 所示。

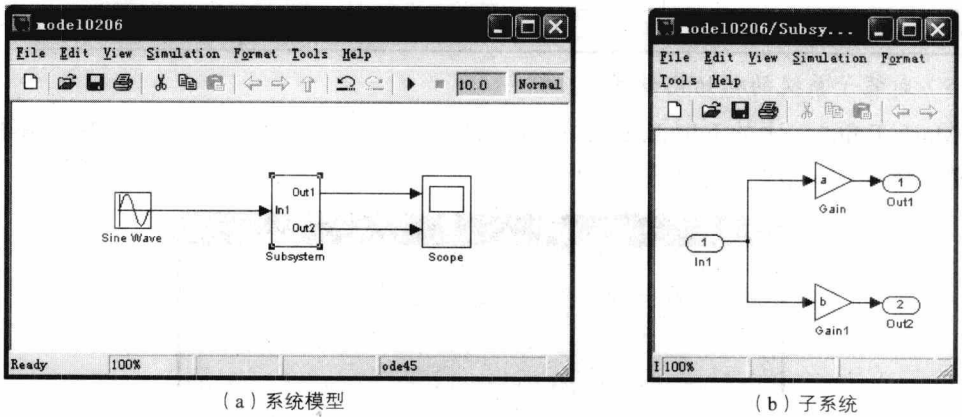
表 2-2　参数设置对话框参数含义

参　数	含　义
Prompt	设置该输入框的名称
Variable	设置变量名，系统将参数输入的值赋给该变量
Type	设置参数输入框的类别，可以有 3 种选择：编辑（edit）、选择框（checkbox）和下拉框（popups）

在参数选项栏中，Popups 下面的输入框为可选择的具体参数值，当输入框为下拉框时，下拉框中将显示 Popups 中的内容。Show parameter 和 Enable parameter 分别表示是否显示该参数输入框和该参数是否可被编辑。在 Callback 栏中可以编写 MATLAB 程序语言，当该参数对话框被选中时，程序被执行。

【示例 2.5】下面构造一个简单的带有参数的子系统，以此示例来讲解参数设置选项卡的具体使用方法。

建立如图 2-47（a）所示的模型，其子系统的内部结构为两个增益模块，如图 2-47（b）所示，增益值分别为变量 a 和 b。对该子系统进行封装并设置参数，使其有一个友好的参数设置界面，可以直接对两个增益模块的参数进行赋值而不必进入到子系统内部分别赋值。其中参数对话框的内容如图 2-48 所示。



（a）系统模型　　（b）子系统

图 2-47　系统模型及其子系统

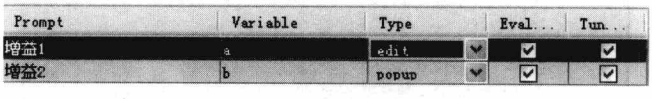


图 2-48　参数对话框的设置

从图中可以看出，第二个参数的输入框类型为 popup，在参数选项的 popup 设置框中输入 3 个值为 1、2、3，注意每行只能输入一个数字。设置完成以后，双击子系统模块，将出现如图 2-49 所示的控制对话框。

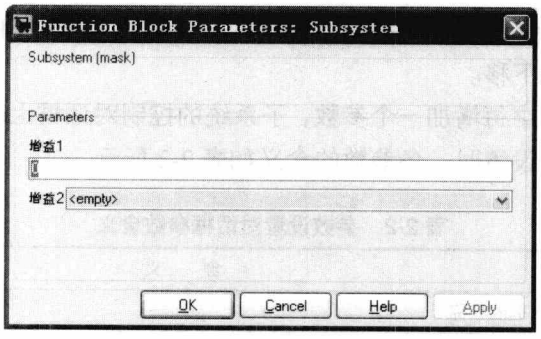


图 2-49　封装子系统的控制对话框

当参数输入框类型为 popup 或 checkbox 时，其返回值将根据是否选中 Evaluate 而不同。

对于 popup 类型，当 Evaluate 被选中时，其返回值为被选择项的序号值。也就是说，当选择了 popup 框中的第 3 个选项时，则将返回 3。而当 Evaluate 未被选中时，其返回值为 popup 选项框中所选中的字符串。对于上面的例子，就是返回字符串“3”。

对于 checkbox 类型，当 Evaluate 被选中时，checkbox 返回 1，否则返回 0。而当 Evaluate 未被选中时，则分别返回“on”和“off”。具体可参照表 2-3。

表 2-3　参数设置表

状　态	选中 Evaluate	为选中 Evaluate
被选中	1	“on”
未选中	0	“off”

（5）封装子系统的初始化设置

“Mask Editor”中的 Initialization 选项卡允许输入 MATLAB 命令，对封装子系统进行一些初始化操作。该选项卡如图 2-50 所示。

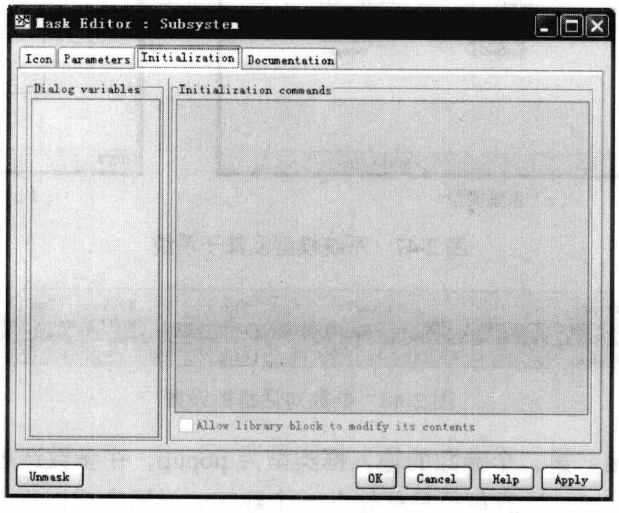


图 2-50　Mask Editor 中的 Initialization 选项卡

Simulink 通常在以下 4 种情况发生时执行命令框中的程序：模型被载入时、仿真开始或模块被更新时、模块被旋转时、模块图标被重绘时（当绘制模块图标需要用到初始化程序所提供的参数时会发生该情况）。

在该选项卡中，左边的 Dialog variables 列表栏列出了封装子系统在参数选项卡中设置的参数变量名。可以通过该列表栏来修改这些变量的名字，也可以将它们复制到命令栏中，方便程序的编写。

右边的 Initialization commands 命令栏，可以在其中写入任何合法的 MATLAB 命令语句。

命令栏中的语句不能访问基础工作空间的变量。并且在具有多条语句时，在每条语句的末尾需要用分号“;”来结束。

在命令栏下方有一个 Allow library block to modify its contents 选项，该选项只对已被加入到模块库中的模块有效。当该选项被选中时，Simulink 允许命令栏中的程序修改模块的参数或增加、删除该模块。也可以通过 sim 命令来使能这个选项。

```
set_param(gcb,'MaskSelfModifiable','on');
```

（6）封装子系统的文档编辑

通过对 Documentation 选项卡的编写，可以定义或修改封装子系统参数界面中的类型名称和模块描述，还能够为该子系统编写帮助文档或将已存在的帮助文档链接到 MATLAB 自身的帮助中去。Documentation 选项卡如图 2-51 所示。

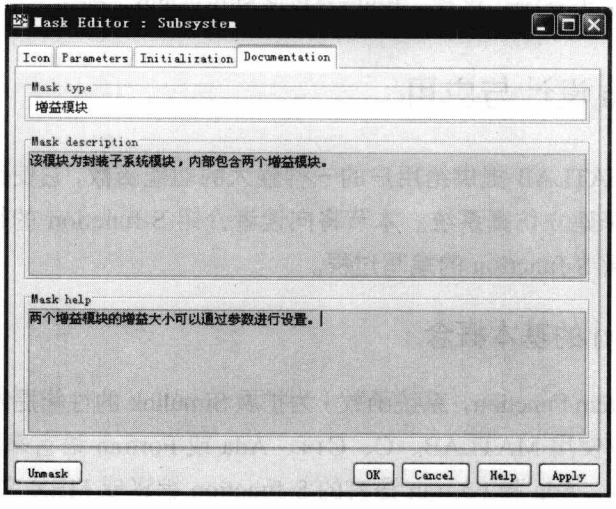


图 2-51　Mask Editor 中的 Documentation 选项卡

选项卡中的第一栏为模块类型栏，可以在其中写入一个该模块的类型名称。中间一栏为对模块的具体描述，如该模块的使用方法等。下面一栏则为模块的帮助文档编辑栏，在其中写入的问题，将被作为该模块的帮助文档加入到 MATLAB 内置的帮助文档中去。

如图 2-51 所示，各个栏中已经填写了相应的内容，其效果可以通过图 2-52 看到。

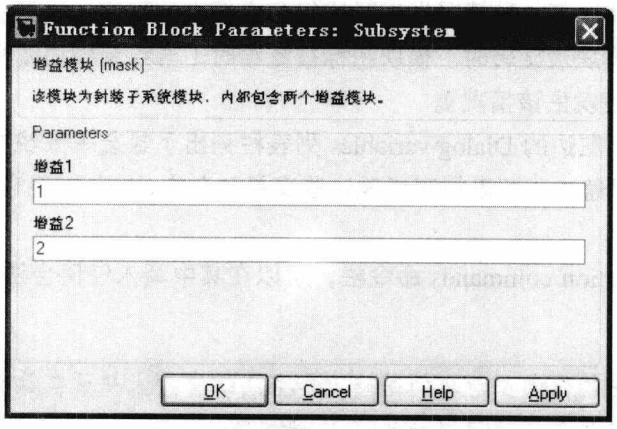


图 2-52　封装子系统模块对话框

对比图 2-51 可以很清楚地看到各栏的作用。单击对话框中的 Help 按钮将出现该模块的帮助文档。

（7）查看封装和解封装

对于一个已封装的子系统要查看其封装前子系统的具体内容，可以选择菜单命令“Edit→Look Under mask”。

若要对已封装的模块进行解封装操作，要先选中此模块，打开封装编辑器“Mask Editor”，单击“Unmask”按钮，则封装就被解开。

若要再次封装该子系统，选择“Edit→Mask Subsystem”即可。

2.4 S-function 设计与应用

S-function 是 MATLAB 提供给用户的一种强大的功能函数，它使得用户可以编写满足自定义功能的模块来建立仿真系统。本节将向读者介绍 S-function 的编写方法，并通过详细的示例让读者了解 S-function 的编写过程。

2.4.1 S-function 的基本概念

S-function（System Function，系统函数）为扩展 Simulink 的性能提供了一个有力的工具。

S-function 可以使用 MATLAB、C、C++、Ada 或 Fortran 语言来编写。使用 MEX 实用工具，将 C、C++、Ada 和 Fortran 语言的 S-function 编译成 MEX 文件，在需要的时候，它们可与其他的 MEX 文件一起动态地连接到 MATLAB 中。

S-function 使用一种特殊的调用格式让你可以与 Simulink 方程求解器相互作用，这与发生在求解器和内置 Simulink 块之间的相互作用非常相似。S-function 的形式是非常通用的，且适用于连续、离散和混合系统。

S-function 为你提供了一种在 Simulink 模型中增加自制块的手段，读者可以使用 MATLAB、C、C++、Ada 或 Fortran 语言来创建自己的块。按照一些简单的规则，读者可

以很方便地在 S-function 中实现自己的算法。在编写一个 S-function 函数，并将函数名放置在一个 S-Functin 块中（在用户定义的函数块库中有效）之后，通过使用菜单命令“Edit →Subsystem”定制用户界面。

读者可以与 Real-Time Workshop（RTW）一起使用 S-function，也可通过编写目标语言编译器（TLC）文件来定制由 RTW 生成的代码。

更多关于 S-function 的信息，读者可以参阅 Workshop 文档资料。

（1）与 S-function 相关的一些基本概念

在编写 S-function 过程中经常会遇到以下几个基本概念，很好地理解它们对于编写 S-function 将会有很大的帮助。这些概念是直接反馈（Direct feedthrough）、可变长度输入（Dynamically sized inputs）以及抽样时间和偏移（Setting sample times and offsets），它们是编写的基础。

- 直接反馈（Direct feedthrough）

直接反馈就是输出（或者可变采样时间模块的可变采样时间）受输入信号的直接控制。简单地说，就是如果输出信号是输入信号的函数，或者在可变步长仿真过程中，S-function 影响着下一个仿真时间的计算，那么这就是直接反馈。例如，$y=k\times u$，其中 u 是输入，k 是增益，y 是输出。

确定一个模块是否存在直接反馈是编写 S-function 的首要任务，因为 Simulink 根据模块中是否存在直接反馈来确定仿真过程中模块的执行顺序。如果模块不是直接反馈型，则计算该模块的输出信号时，这个模块由于不需要等待前一个模块的输入而有可能先于前面的模块执行。

- 可变长度输入（Dynamically sized inputs）

S-function 支持任意长度的输入信号。其过程就是在仿真开始的时候计算输入信号向量维数，从而动态地确定具体的输入信号长度，继而驱动 S-function。而且输入信号维数也可被用来决定连续状态个数、离散状态个数和输出的信号个数。

用 M 文件编写的 S 函数只允许有一个输入端口和输出端口，其信号长度可以是动态确定的。而 C 语言 S-function 则可以有多个输入/输出端口，同时，每个端口的维数也可以是动态确定的。因此相比之下，C 语言 S-function 具有更好的灵活性。

如果 M 文件 S-function 把它的连续、离散状态或者输出信号设置为可变长度，则 Simulink 会根据输入信号的长度来确定它们的长度，并且与输入信号长度相等。

- 抽样时间和偏移（Setting sample times and offsets）

M 文件 S-function 和 C 语言 S-function 在运行时间设置方面都有很强的灵活性，其抽样时间可以是连续的、离散的或者固定的。总之，Simulink 为采样时间提供以下选择：

① 连续采样时间

对于 S-function 来说，连续采样时间包括连续状态和非采样过零点。在这种情况下，输出按最小时间步改变。

② 连续但固定最小步长采样时间

在这种时候，S-function 需要在每个主仿真时间步运行，但是在最小仿真步内值不改变。

③ 离散采样时间

如果用户的 S-function 模块的行为是在离散的时间间隔发生的，那么，用户可以定义一个抽样时间来决定 Simulink 调用该模块的时间。用户还可以定义一个偏移时间来延迟每一个采样点。但是偏移时间的值不可以超过系统的采样时间。

采样点时间由下面公式决定：

$$\text{TimeHit} = (n * \text{period}) + \text{offset} \qquad (式\ 2\text{-}1)$$

其中 n 是整数，为当前仿真步，初始值总是 0。

如果用户定义了离散采样时间，则 Simulink 会在每个采样点调用 mdlOutput 和 mdlUpdate。

④ 可变采样时间

可变采样时间就是指相邻采样时间间隔可变的离散抽样时间。在每个仿真步的开始，可变采样时间的 S-function 需要首先计算下一个采样点的时刻。

⑤ 继承采样时间

有时，S-function 模块没有固定的采样时间特性，这时用户就可以把这个模块的采样时间设置为继承型。这样，这个模块的本身状态是连续的还是离散的就取决于系统中的其他模块了。

一个模块可以有以下几种方式来继承采样时间：继承驱动模块的采样时间、继承目标模块的采样时间、继承系统中最快的采样时间。

S-function 可以是单速率或者多速率，对于多速率的 S-function 则有多个采样时间。此时，S-function 的抽样时间是一个具有两列元素的矩阵，其中每一行的元素表示一种抽样速率。

（2）S-function 工作的基本原理

用户在自己编写 S-function 之前，了解 S-function 工作的基本原理是相当有必要的，这对于理解 Simulink 的整个仿真原理也很有好处。因此，我们先简单地介绍一下 S-function 的基本工作原理。

• S-function 的数学模型

Simulink 模块一共由 3 部分组成：输入变量、状态变量和输出变量。其中输出变量又是抽样时间、输入变量和状态变量的函数。如图 2-53 所示。

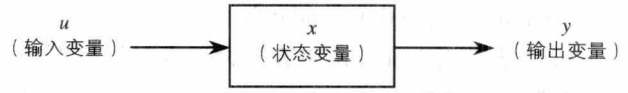


图 2-53　Simulink 模块的基本元素

输入变量、状态变量和输出变量这三者之间的关系由下列等式给出，即：

$$y = f_0(t, x, y) \qquad (式\ 2\text{-}2)$$

$$\dot{x} = f_d(t, x, y) \qquad (式\ 2\text{-}3)$$

$$x_{d_{k+1}} = f_u(t, x, y) \qquad (式\ 2\text{-}4)$$

- 仿真阶段

Simulink 模块在仿真时的处理过程是按阶段进行的。首先是初始化阶段，此时 Simulink 将库模块与模型融为一体，初始化输入/输出宽度、数据类型、抽样时间、估算模块参数、决定模块的执行顺序以及分配内存。然后 Simulink 才进入仿真循环，其中经过的每一个小循环被称为一个仿真步。在每个仿真步里，Simulink 按照初始化阶段决定的顺序执行模型里面的每一个模块。对每一个模块来说，Simulink 都要为当前时刻回调计算，更新模块状态函数并进行输出，这个过程将一直持续到仿真结束。对于连续状态求解器，一个仿真步长又分为主时间步（major time step）和最小时间步（mionr time step），其中最小时间步是主时间步的一部分，Simulink 在最小时间步做数值积分运算。图 2-54 有两个输出环节和求微分环节，为了提高积分精度，求解器会进行两次输出一致性检查，当有两个输出大于求解器页面所设置的误差限制时，会以一个小的步长重新计算输出和微分。每个仿真循环在最后检查是否有过零事件的发生，一旦检测到过零事件，Simulink 在发生过零事件的变量或者状态的当前值和一个仿真步长之前的值之间进行插值运算，以提高仿真精度。

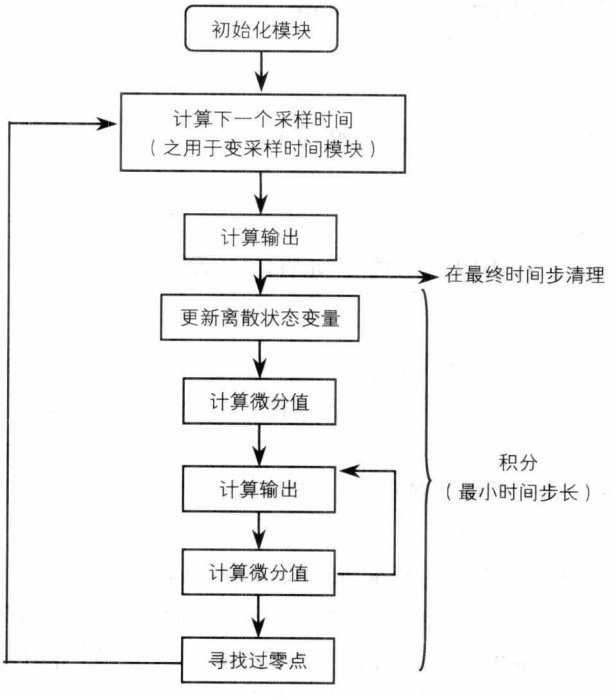


图 2-54　仿真流程图

- S-function 的回调方式

S-function 依靠回调函数完成每个仿真阶段的任务。每个模型的仿真期间，在每个仿真阶段，Simulink 调用模块中每个 S-function 模块相应的函数。S-function 里调用的函数所

完成的任务包括以下几个方面。

① 初始化

在仿真循环开始前，Simulink 初始化 S-function。在这个阶段，Simulink 将完成以下完成几个工作：初始化结构体 SimStruct，这个仿真结构体包含了 S-function 的所有信息；设置输入/输出端口的大小和维数；设置模块的抽样时间；分配存储空间。

② 计算下一个采样点时间

如果是可变抽样时间模块，则这个阶段计算下一个抽样点的时间，也就是说计算下一步的仿真步长。

③ 计算主时间步的输出

在这个函数调用结束后，此模块的所有输出端口对当前时间步都是有效的。

④ 在主时间步更新离散状态

在调用这个函数时，所有的模块都应该执行单步单次的工作，例如，在主仿真循环里为下一个时间更新离散状态。

⑤ 积分计算

这一步只有连续状态或带有非采样过零点时才有效。Simulink 以较小时间步长调用输出和微分函数。正因为如此，Simulink 才可以计算 S-function 的状态。如果 S-function 含有非采样过零点（仅当 C MEX 的情况），Simulink 才会以较小时间步来计算 S-function 的输出和过零点部分，从而确定过零点的位置。

• S-function 的实现

要实现 S-function 可以用 M 文件或者 MEX 文件。接下来的章节将简要介绍这两种实现方法以及讨论各自的优缺点。

① M 文件的 S-function

一个 M 文件的 S-function 由以下形式的 MATLAB 函数组成：

```
[sys, x0, str, ts] = f(t, x, u, flag, p1, p2, ...)
```

其中，f 是 S-function 的名字；t 是仿真时间的当前值；x 是相应 S-function 模块的状态向量；u 是模块的输入；flag 是仿真流程标志向量。另外，Simulink 还可以在仿真过程中向 S-function 传递一些额外的参数 p1、p2、…，它们可以用于 S-function 的各个计算过程中。每次 S-function 执行完一项任务的时候，它将以一个结构的形式返回结果，这个结构的具体形式可以参照语法示例。

在目录“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”里可以找到 M 文件 S-function 的模块。这个模板由头文件和一系列子函数组成。头函数激活了 flag 要标示调用的子函数。而子函数也被称为 S-function 的回调函数，在仿真期间执行 S-function 所要求的任务。表 2-4 列出了 M 文件 S-function 的各部分内容，以及它们需要遵循的标准形式。

表 2-4 M 文件 S-function 各部分内容

仿真阶段	S-function 程序	flag 值
初始化模块阶段	mdlInitializeSizes	flag = 0
计算下一个采样时间（只用于变采样时间模块）	mdlGetTimeOfNextVarHit	flag = 4

续表

仿真阶段	S-function 程序	flag 值
计算输出	MdlOutputs	flag = 3
更新离散状态	mdlUpdate	flag = 2
计算微分值	mdlDerivatives	flag = 1
仿真任务结束	mdlTerminage	flag = 9

当读者编写 M 文件 S-function 时，最好遵循这个结构以及模板里的命名习惯。这样做在方便他人理解的同时也方便自己维护。

② MEX 文件

类似于 M 文件的 S-function，MEX 文件 S-function 也由一系列的回调函数组成，这些函数 Simulink 将在仿真期间的不同阶段被相应地调用。然而，M 文件的 S-function 和 MEX 文件的 S-function 存在着重要的不同点。一方面，MEX 文件 S-function 是以不同的编程语言编写的：C、C++、Ada 和 Fortran。而且，Simulink 直接调用 MEX 文件 S-function 函数的程序不是像 M 文件 S-function 那样要使用 flag 值。由于 Simulink 直接调用函数，所以 MEX 文件 S-function 必须遵守 Simulink 规定的命名习惯。另外，MEX 文件 S-function 可调用的函数比 M 文件的多得多。MEX 文件 S-function 还可以直接访问内部的数据结构，就是所谓的 SimStruct，Simulink 用这个结构来保存有关这个 S-function 的信息。MEX 文件的 S-function 还可以用 MATLAB MEX 文件 API 来直接访问 MATLAB 的工作空间。

C 语言 MEX 文件 S-function 模板存放于“MATLAB 根目录/matlabroot/toolbox/simulink/src”路径下面，名字为 sfuntmpl_basic.c。这个模板包含必需的以及可选的 C MEX 文件 S-function 可以实现的回调函数的执行框架。如果想了解详细的模板结构，可以在同一个目录下参看 sfuntmpl_doc.c。

③ MEX 文件 S-function 与 M 文件 S-function 的对比

MEX 文件 S-function 与 M 文件 S-function 各有优缺点。M 文件 S-function 的开发速度较快，它不需要编译链接执行循环的时间，而且可以更方便地访问 MATLAB 和 toolbox 函数。MEX 文件 S-function 的主要优点就是相当的灵活多变。它拥有更多的回调函数而且可以访问 SimStruct，这使得它可以实现比 M 文件 S-function 更灵活多变的功能，包括可以处理除双精度、复数和矩阵输入以外的数据类型。

2.4.2　在模型中使用 S-function

为了能在 Simulink 中使用 S-function，必须从 Simulink 中的 User-Defined Functions 模块库中向 Simulink 模型文件窗口中拖放 S-function 模块。然后在 S-function 模块的对话框中的 S-function names 框中输入 S 函数的文件名。

【示例 2.6】 调用 S-function。

具体操作步骤如下：

① 建立模型，如图 2-55 所示；

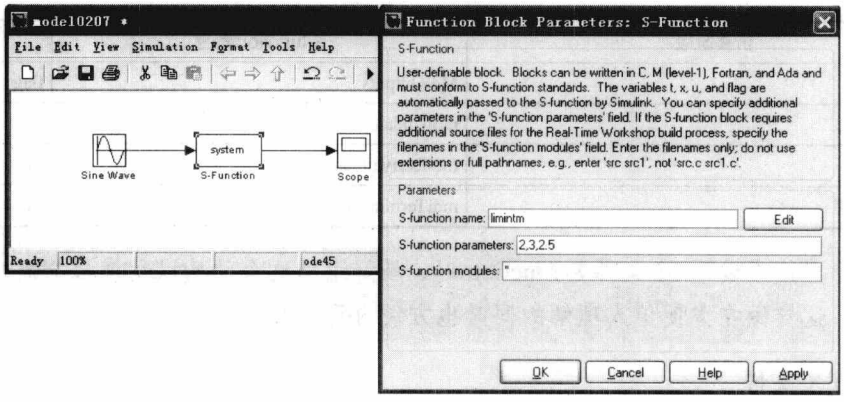


图 2-55　设置参数的 S-function 模型

② 双击 S-function 模块，弹出参数对话框，在 S-function Names 文本框中输入 limintm，在 S-function parameters 文本框中输入“2，3，2.5”，然后单击“OK”按钮；

③ 运行 Simulink 模型；

④ 结果如图 2-56 所示。

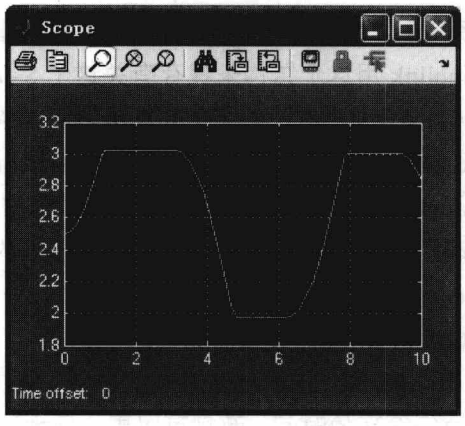


图 2-56　运行结果

这个示例使用了 limintm 函数，是一个调用采样的 S-function Simulink 模型，函数的原代码在“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”文件夹中。limintm 函数接受 3 个参数：上界、下界和初始值。如果积分在上界与下界之间，则它的输出是输入信号的积分，如果积分值大于上界，或者小于下界，那么就分别保持上、下界值。此例中下界、上界和初始条件分别是 2、3 和 2.5。

系统自带了 limintm 程序原代码，用户可以直接在参数对话框中添加，具体程序及说明如下：

```
function [sys,x0,str,ts]=limintm(t,x,u,flag,lb,ub,xi)
%LIMINTM 执行受积分限制
%    是一个连续 M 文件的 S-function 执行一个连续受限积分的例子
%    其中输出受到上界（UB）和下界（LB）的限制
```

```
%   系统的初始值为 (XI)
%
%   关于 S-function 模板可参考 sfuntmpl.m.
%
%

switch flag

  %%%%%%%%%%%%%%%%%%
  % 初始化 %
  %%%%%%%%%%%%%%%%%%
  case 0
    [sys,x0,str,ts] = mdlInitializeSizes(lb,ub,xi);

  %%%%%%%%%%%%%%%
  % 求导 %
  %%%%%%%%%%%%%%%
  case 1
    sys = mdlDerivatives(t,x,u,lb,ub);

  %%%%%%%%%%%%%%%%%%%%%%%
  % 更新 %
  %%%%%%%%%%%%%%%%%%%%%%%
  case {2,9}
    sys = []; % do nothing

  %%%%%%%%%%
  % 输出 %
  %%%%%%%%%%
  case 3
    sys = mdlOutputs(t,x,u);

  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end

% limintm 结束

%
%=============================================================================
% mdlInitializeSizes
% 向 S-function 返回大小、初始条件和样本时间
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(lb,ub,xi)

sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
```

```
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
str = [];
x0  = xi;
ts  = [0 0];   % sample time: [period, offset]

% mdlInitializeSizes 结束

%
%=============================================================
% mdlDerivatives
% Compute derivatives for continuous states.
%=============================================================
%
function sys = mdlDerivatives(t,x,u,lb,ub)

if (x <= lb & u < 0)  | (x>= ub & u>0 )
  sys = 0;
else
  sys = u;
end

% mdlDerivatives 结束

%
%=============================================================
% mdlOutputs
% S-function 输出
%=============================================================
%
function sys = mdlOutputs(t,x,u)

sys = x;

% mdlOutputs 结束
```

最常使用的就是用 S-function 来创建一个用户自定义的 Simulink 模块。除此之外，还可以用 S-function 来实现以下几个方面的功能：

- 向 Simulink 模型中增加一个通用目的的模块；
- 使用 S-function 的模块来充当硬件的驱动；
- 在仿真系统中嵌入已经存在的 C 代码；
- 将系统表示成一系列的数学方程；
- 在 Simulink 中使用动画。

使用 S-function 的一个优点就是，用户可以构建一个通用目的的模块，在一个模型中可以多次使用，每一个模块可以有不同的参数。

2.4.3　M 文件 S-function 的编写

M 文件 S-function 是采用 MATLAB 语言编写的，它采用 MATLAB 中 M 文件的语法结构，编写代码之后不需要编译就可以直接使用，因而它更加便于编写和使用。在此用一个简单的示例来引导读者进行深入的学习。

（1）M 文件 S-function 模板

第 2.4.2 节给出了一个简单的具体示例，本节将详细介绍 M 文件 S-function 模板。Simulink 给出了一个 M 文件 S-function 模板 sfuntmpl.m，该文件存放在“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”目录下，它提供了 M 文件 S-function 的基本框架，通过它可以快速地编写函数代码。下面结合 M 文件 S-function 模板代码及其流程图(如图 2-57 所示) 来说明它的构成及具体实现。

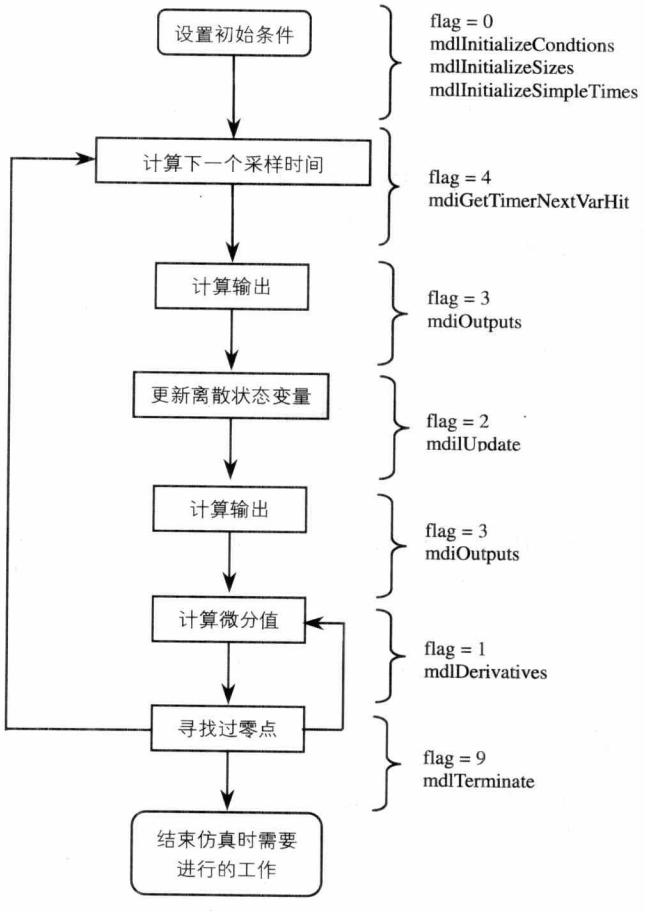


图 2-57　M 文件 S-function 模板流程图

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
% M文件S-function主体部分
% 函数名称: sfuntmpl
```

```
% 主要功能：根据输入参数 flag 的数值调用相应的函数
switch flag,

  %%%%%%%%%%%%%%%%%%
  % 初始化 %
  %%%%%%%%%%%%%%%%%%
  case 0,
  %当 flag = 0 时调用 mdiInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes;

  %%%%%%%%%%%%%%%%
  % 求导 %
  %%%%%%%%%%%%%%%%
  case 1,
  % 当 flag = 1 时调用 mdlDervatives 函数计算连续状态的数量
    sys=mdlDerivatives(t,x,u);

  %%%%%%%%%%
  % 更新 %
  %%%%%%%%%%
  case 2,
  % 当 flag = 2 时调用 mdlUpdate 函数计算离散状态的数值
    sys=mdlUpdate(t,x,u);

  %%%%%%%%%%%
  % 输出 %
  %%%%%%%%%%%
  case 3,
  % 当 flag = 3 时调用 mdlOutputs 函数计算输出信号的数值
    sys=mdlOutputs(t,x,u);

  case 4,
  % 当 flag = 4 时调用 mdlGetTimeOfNextVarHit 函数计算下一个抽样时刻
    sys=mdlGetTimeOfNextVarHit(t,x,u);

  case 9,
  % 当 flag = 9 时调用 mdlTerminate 函数结束仿真
    sys=mdlTerminate(t,x,u);

  otherwise
  % 当 flag 为其他数值时表示仿真过程出错
    error(['Unhandled flag = ',num2str(flag)]);

end

% sfuntmpl 结束

%
% mdlInitializeSizes
% S-function 的初始化.
```

```
%
function [sys,x0,str,ts]=mdlInitializeSizes

%
% 调用 simsizes 获得一个用于存放长度信息的结构
% -1 表示动态确定的范围
sizes = simsizes;
% 设置连续状态个数
sizes.NumContStates = 0;
% 设置离散状态个数
sizes.NumDiscStates = 0;
% 设置输出信号个数
sizes.NumOutputs = 0;
% 设置输入信号个数
sizes.NumInputs = 0;
% 设置直接反馈状态
% 0 表示没有直接反馈
% 1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间个数最少为 1
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
% 设置 S-function 的初始状态 x0
x0  = [ ];
% 设置 S-function 的保留参数 str ( 应该设置为空向量 )
str = [ ];
%
% 抽样时间
% ts 是一个具有两列元素的矩阵，其中第一列表示抽样时间，另外一列表示时间偏移
ts  = [0 0];
% mdlInitializeSizes 结束

%
% mdlDerivatives
% 计算 S-function 连续状态的导数，返回给 Simulink 进行积分计算
%
function sys=mdlDerivatives(t,x,u)
% 计算 S-function 连续状态的导数并通过 sys 参数返回给 Simulink.
sys = [ ];

% mdlDerivatives 结束

%
% mdlUpdate
% 计算 S-function 的离散状态并向 Simulink 返回这些状态的数值
% requirements.
%
function sys=mdlUpdate(t,x,u)
```

```
%计算 S-function 的离散状态并通过 sys 参数返回给 Simulink
sys = [ ];

% end mdlUpdate

%
% mdlOutputs
% 计算 S-function 的输出信号并返回给 Simulink 作为模块的输出.
%
function sys=mdlOutputs(t,x,u)
% 计算 S-function 的输出信号并通过 sys 参数返回给 Simulink.
sys = [ ];

% mdlOutputs 结束

%
% mdlGetTimeOfNextVarHit
% 计算下一个抽样时刻并且返回给 Simulink
% 本函数只适用于可变步长离散仿真时间且在初始化过程中把 ts 设置为[-2 0]
%
function sys=mdlGetTimeOfNextVarHit(t,x,u)
% 一个设置下一个抽样时刻的示例
% 下一个抽样时刻设置为与当前时刻相差 1s
sampleTime = 1;
sys = t + sampleTime;

% mdlGetTimeOfNextVarHit 结束

%
% mdlTerminate
% 在仿真结束时执行清理工作，如回收内存等
%
function sys=mdlTerminate(t,x,u)
% 设置返回参数 sys 为空矩阵[ ]
sys = [ ];
% mdlTerminate 结束
```

S-function M 文件在运行过程中总要检验输入变量 flag 的值，然后按照一定的规则执行相应的操作。Math Works 样板中采用的方法是依据 flag 的值来决定是否调用内部函数的 switch...case 模块。如前所述，变量 flag 一共有 6 种可能值，接下来将具体讨论这 6 种可能值所对应的操作。

- 初始化（flag = 0）

S-function 的初始化一共包括 3 个步骤：

① 设置 sizes 结构，并将此结构分配到多目标返回变量 sys 中去。

使用下面的语句可以添加一个 sizes 结构：

```
sizes = simsizes;
```

这一语句返回一个没有初始化的 sizes 结构，然后对 sizes 结构的每个元素赋值。sizes 结构是 S-function 的信息载体，它包含 6 个变量，开始时都分别赋予初值 0。它们具体的说明信息参见 S-function 模板。sizes 结构的每个元素必须有值，即使为 0。

② 设置初始化条件向量 x0。

例如，有两个连续状态，初始变量为 2.0，有两个离散状态，初始变量为 0.0，则应按照下式进行设置：

```
x0 = [2.0 2.0 0.0 0.0];
```

③ 设置 str 变量。

创建采样时间及延迟矩阵 ts。对 ts 的每个采样时间必须是在同一行内，甚至对于连续 S-function 也至少有一行。如果采样时间设置为-1，则采样时间将从与 S-function 模块输入相联系的模块中继承下来。

- 连续状态微分（flag = 1）

如果 flag = 1，则将状态向量连续部分的微分值赋给 sys。注意，如果同时存在离散状态，则 sys 的变量将与 x 的不同，因为 x 同时包括着连续和离散状态。

若由某 S-function 模拟下面的非线性系统：$\dot{x} = x_2$，$\dot{x}_2 = x_1 + \dot{x}_2 + u_1$，则用下面的语句来设置 sys 属性：

```
sys(1) = x(2);
sys(2) = x(1)+x(2)^2+u(1);
```

- 离散状态更新（flag = 2）

如果 flag = 2，则将状态向量离散部分的微分值赋给 sys。如果有多个采样时间（包括含有采样时间为 0 的连续状态的混合系统），则 S-function 首先要检验当前时刻是否有采样任务以及有哪个状态需要采样，然后再更新采样状态的值。注意，状态向量中的所有元素必须有值。

例如，有如下一阶变量离散子系统：

$x_1(k+1) = x_1(k) + u_1(k)$

若 flag = 2，则使用下列语句来设置 sys：

```
sys = x(1)+u(1);
```

现在来考虑有两个采样时间的二阶离散子系统。第 1 个采样时间为 0.3 秒，第 2 个为 0.5 秒，而且都没有时延，则第 1 个状态的更新将根据下面的方程进行：

$$x_1(k+1) = x_1(k) + 0.5x_2(k)$$

下面的语句用来检验仿真时间，并在每次采样时间达到的时候更新 sys。

```
period_1 = 0.3;
offset_1 = 0.0;
period_2 = 0.5;
offset_2 = 0.5;
sys = x;
```

```
if abs(round(t-offset_1)/period_1-((t-offset_1)/period_1))<1.0e-8)
   Sys(1) = sys(1)+0.5*x(2);
end
if abs(round(t-offset_2)/period_2-((t-offset_2)/period_2))<1.0e-8)
   sys(2) = sys(2)+u(1)
end
```

• 模块输出（flag = 3）

如果 flag = 3，则将 S-function 的输出赋给 sys。

• 下次采样时间（flag = 4）

如果 flag = 4，则将 sys 的值赋给下一次采样时刻的值，只有在采样时刻可变时，S-function 才会以 flag = 4 的条件被调用。

• 仿真结束（flag = 9）

当仿真因某种原因结束时，函数将以 flag = 9 被调用。S-function 此时需要完成一些终止仿真的任务，不需要给 sys 赋值。

（2）M 文件 S-function 示例

学习编程的最简单和最直接的方法就是读程序，编写 S-function 也不例外。在本小节里，我们将通过几个示例程序来学习 M 文件 S-function 的设计方法。这些示例部分取自 MATLAB 自带的程序，内容具有很广泛的代表性，更重要的是这些例子都简单易懂，是 S-function 初学者的经典素材。每个示例程序都详细地介绍了程序的结构和功能，并且列出了程序代码，同时在程序段中还添加了注释，详细介绍了各个代码段的功能。

• 连续状态系统

下面通过一个示例程序来说明连续状态系统的 M 文件 S-function 的编写。这个示例程序是 MATLAB 自带的，位于“MATLAB 根目录/matlabroot/toolbox/simulink/simdemos/simfeatures”中的 sfcndemo_csfunc.mdl 文件。

【示例 2.7】连续状态系统 S-function 示例。图 2-58 是该模型的框图。

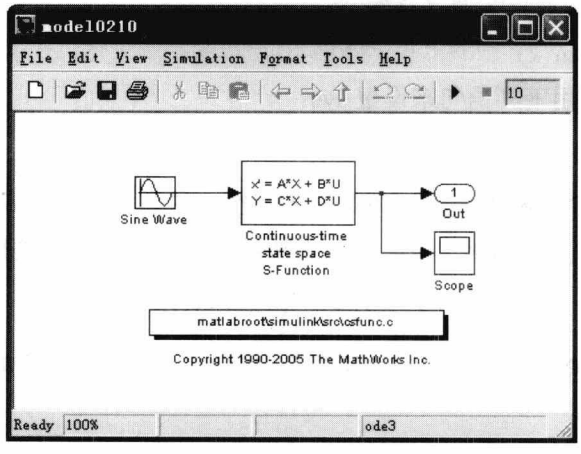


图 2-58　一个连续状态系统示例

在这个示例程序中，正弦信号产生器（Sine Wave）产生一个具有两个元素的正弦信号

向量，向量中的元素具有连续的抽样时间，信号幅度分别是 1 和 2。而 Continuous-time state space S-function 模块对输入信号进行变换，其输出通过 Scope 模块显示出来。

该连续状态模块由 M 文件 S-function csfunc.m 构造，它位于“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”中。

该模块对输入信号向量实施的矩阵变换如下式，即：

$$\begin{cases} x' = \boldsymbol{A}x + \boldsymbol{B}u \\ y = \boldsymbol{C}x + \boldsymbol{D}u \end{cases} \qquad (式\ 2\text{-}5)$$

其中 $\boldsymbol{A} = \begin{pmatrix} -0.09 & -0.01 \\ 1 & 0 \end{pmatrix}$；$\boldsymbol{B} = \begin{pmatrix} 1 & -7 \\ 0 & -2 \end{pmatrix}$；$\boldsymbol{C} = \begin{pmatrix} 0 & 2 \\ 1 & -5 \end{pmatrix}$；$\boldsymbol{D} = \begin{pmatrix} -3 & 0 \\ 1 & 0 \end{pmatrix}$。

把 M 文件和 mdl 文件放置于相同目录下之后便可以启动仿真程序，程序结束后可以得到示波器的输出，如图 2-59 所示。

由图 2-59 可以看出输出信号是连续的。

- 离散状态系统

下面通过一个示例程序来说明离散状态系统 M 文件 S-function 的编写。这个示例程序是 MATLAB 自带的，在“MATLAB 根目录/matlabroot/toolbox/simulink/simdemos/simfeatures”目录下的 sfcndemo_dsfunc.mdl 文件中。

【示例 2.8】离散状态系统 S-function 示例。图 2-60 是该模型的框图。

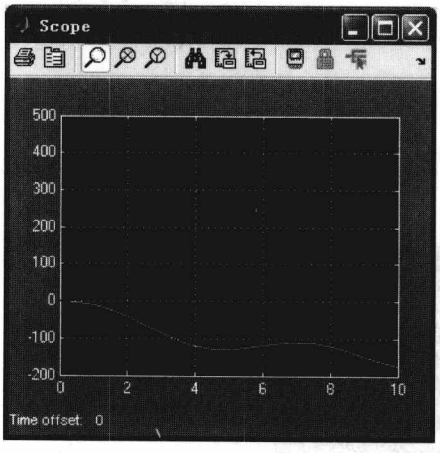


图 2-59　连续状态系统的示例程序仿真结果

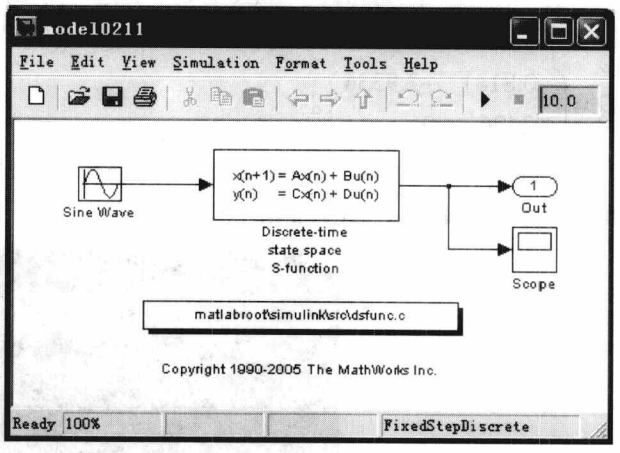


图 2-60　一个离散系统的示例程序

在这个示例程序中，正弦信号产生器产生一个具有两个元素的正弦信号向量，向量中的元素具有离散的抽样时间，信号幅度分别是 1 和 2。而 Discrete-time state space S-function 模块对输入信号进行变换，然后其输出通过 Scope 模块显示出来。

该离散状态模块由 M 文件 S-function dsfunc.m 构造，它位于“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”。在本示例中使用的 S-function 模块设置如图 2-61 所示。

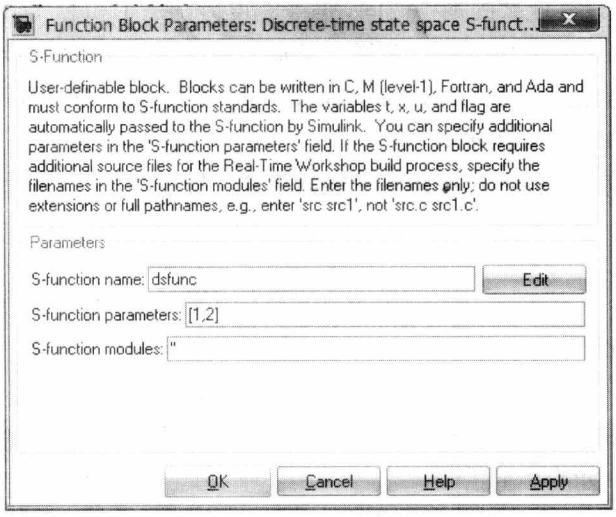


图 2-61　系统仿真时间参数设置

该模块对输入信号向量实施矩阵变换如下式，即：

$$\begin{cases} x(n+1) = \boldsymbol{A}x(n) + \boldsymbol{B}u(n) \\ y(n) = \boldsymbol{C}x(n) + \boldsymbol{D}u(n) \end{cases} \quad (式\ 2\text{-}6)$$

其中 $\boldsymbol{A} = \begin{pmatrix} -1.3839 & -0.5097 \\ 1.0000 & 0 \end{pmatrix}$；$\boldsymbol{B} = \begin{pmatrix} -2.5559 & 0 \\ 0 & 4.2382 \end{pmatrix}$；$\boldsymbol{C} = \begin{pmatrix} 0 & 2.0761 \\ 1 & 7.7891 \end{pmatrix}$；$\boldsymbol{D} = \begin{pmatrix} -0.8141 & -2.9334 \\ 1.2426 & 0 \end{pmatrix}$。

把 M 文件和 mdl 文件放置于相同目录下之后便可以启动仿真程序，程序结束后可以得到示波器的输出，如图 2-62 所示。

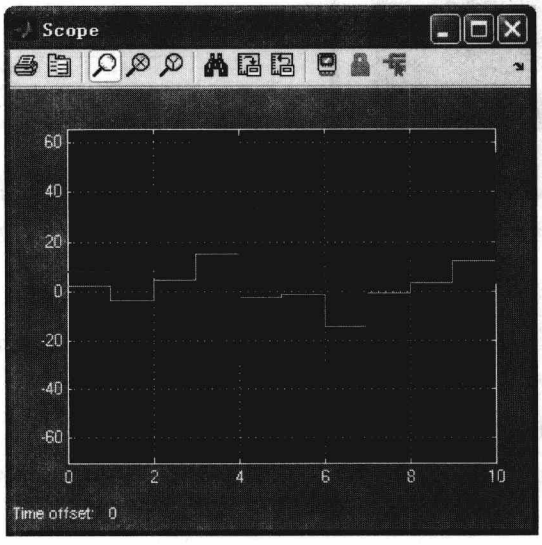


图 2-62　离散状态系统的示例程序仿真结果

由图 2-62 可以看出输出信号是离散的。

- 混合状态系统

前面介绍了连续和离散系统的 M 文件 S-function 的编写方法。有时系统中同时存在连续状态和离散状态。

【示例 2.9】本示例是由一个 Integrator（积分器）和一个 Unit Delay（单位时延模块）组成的，输入信号首先通过积分器进行积分，然后通过单位时延模块把信号延时一个单位之后进行输出，如图 2-63 所示。

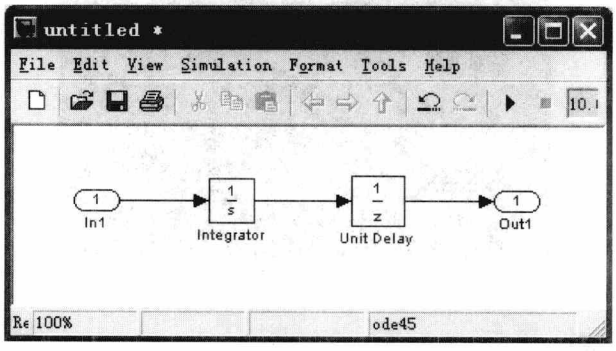


图 2-63　混合系统的构成

这个示例程序是 MATLAB 自带的，是“MATLAB 根目录/matlabroot/toolbox/simulink/simdemos/simfeatures”中的 sfcndemo_mixedm.mdl 文件。图 2-64 是该模型的框图。

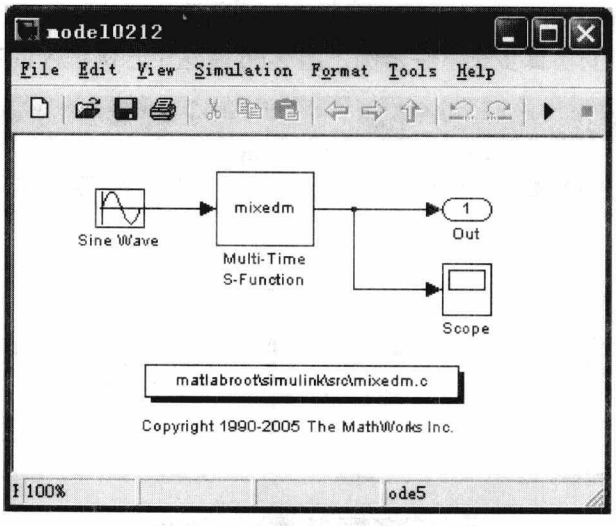


图 2-64　一个混合系统的示例图

在这个示例程序中，正弦信号产生器产生一个幅度为 1 的正弦信号向量，然后通过混合状态模块对这个信号进行积分和延迟，其输出通过 Scope 模块显示出来。

该混合状态函数由 M 文件 S-function mixedm.m 构造，它位于“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”中。在本示例中使用的 S-function 模块。

如图 2-65 所示，容易看出仿真结束的输出信号是一个周期为 1 的离散信号，它是对连续的积分信号实施抽样和保持的结果。

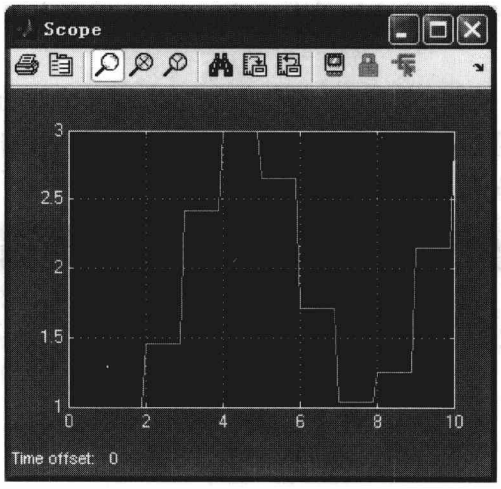


图 2-65　混合系统示例程序的运行结果

• 可变仿真步长系统

前面曾经提到，S-function 的抽样时间是可以变化的，Simulink 通过调用相应的函数来确定下一个仿真时刻。下面将通过一个 MATLAB 示例程序 sfcndemo_vsfunc.mdl 来说明可变仿真步长 M 文件 S-function 的设计方法，它存放在“MATLAB 根目录/matlabroot/toolbox/simulink/simdemos/simfeatures”目录下。

【示例 2.10】可变仿真步长系统示例。图 2-66 是这个示例程序的系统框图。

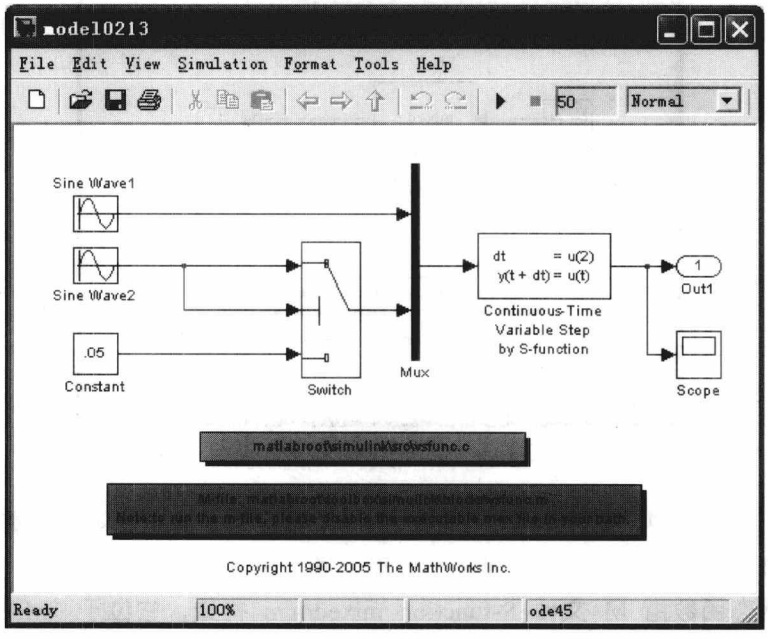


图 2-66　一个可变仿真步长系统的示例

可变步长模块（Continuous-Time Variable Step by S-function）的输入信号是由两个信号复用而成的，其中第 1 个输入信号是一个正弦信号，第 2 个输入信号是一个截掉后的正弦信号，它是由一个幅度为 2 的正弦信号通过一个选通器模块（Switch）之后得到的。

在此示例中，两个正弦信号产生器产生幅度分别为 1 和 2 的正弦信号，第 2 个正弦信号产生器通过一个选通器模块，这个选通器模块有 3 个输入端口，当第 2 个输入端口的输入信号大于第 1 个预先设定的数值时，选通器输出第 1 个输入端口的信号，否则，输出信号等于第 3 个输入端口的数据。

第 1 个正弦信号发生器和选通器模块的输出信号通过复用器（Mux）复合成一个信号进入 Continuous-Time Variable Step by S-function，产生的输出信号通过示波器模块显示。

该模块由 M 文件 S-function vsfunc.m 构造，它位于“MATLAB 根目录/matlabroot/toolbox/simulink/blocks”目录下。

如图 2-67 所示是仿真结果，从中可以看到一些比较尖锐的信号幅度变化，这是由于第 2 个正弦信号的幅度低于 0.1 时，选通器的输出信号为 0.05，可变步长模块的抽样周期变为 0.05 秒，从而造成比较尖锐的幅度变化。当第 2 个正弦信号的幅度大于 0.1 时，选通器的输出信号就等于这个正弦信号的幅度，这时可变步长模块的抽样周期取决于第 2 个正弦信号的幅度。

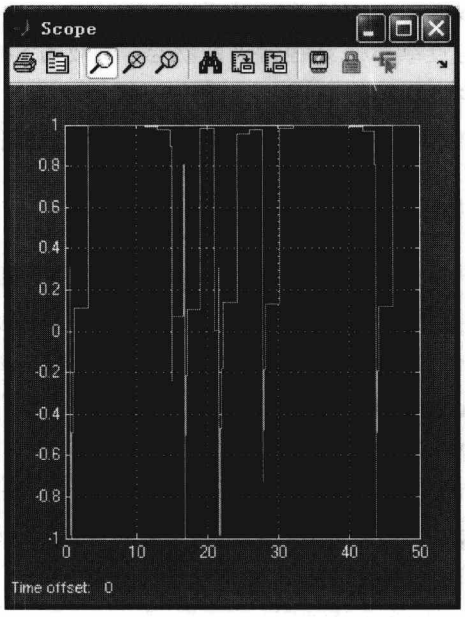


图 2-67　可变仿真步长系统的运行结果

- M 文件 S-function 编程注意

① M 文件 S-function 输入/输出都是数据流格式。

② M 文件 S-function 输入参数可以是复数、矩阵。

③ M 文件 S-function 输入/输出都只能是实数。

④ M 文件 S-function 输入/输出端口都只有一个。

⑤ M 文件 S-function 输入/输出不能是矩阵，不能动态定大小。

⑥ 局部变量的存储。

由于 S-function 需要频繁地进行局部变量的存储，而且它又是 MATLAB 的函数型 M 文件，所以在 S-function 调用时，读取局部变量必须要进行初始化。使用全局变量倒是个可行的方法，但是却不是推荐使用的方法，因为在仿真时，此变量所占用的内存将被始终占用，而不可以被多次共享，极大地浪费了资源。在此介绍一种比较理想的方法，就是使用 S-function 模块的 UserData 函数。UserData 可以是一个标量、矩阵、单元数组或结构，因此，对其所能存储的数据没有数量和类型的限制。

例如，若一 S-function 需要存储时间向量和状态向量，以便下次 S-function 调用时使用，可以使用下面的语句：

```
u_dat.time = t;
u_dat.state = x;
set_param(gcb,'UserData',u_dat);
```

⑦ 动态尺寸。

如果将输入个数设置为-1，表示输入个数根据其前一个模块的输出而定，输出设置为-1 则表示输出个数和输入个数相同，也是动态的。

⑧ 混合系统 S-function。

S-function 可以既包含连续状态又包含离散状态。离散的 S-function 可以有多个采样时间。在很多情况下，这种功能都很有用。但是在一般情况下，还是推荐使用建立单目标 S-function。因为这会使得更易扩展和维护并且能更方便地被其他对象重复使用。

2.4.4 C 语言 S-function 的编写

C 语言 S-function 的代码采用通用的 C 语言规范，因而它同时具有 C 语言和 S-function 的特点，它更适合于设计通用的 S-function 模块，它比 M 文件 S-function 具有更强的处理能力，能够更好地实现对仿真过程的控制，另外，利用 Simulink 中的 S-function Builder 可以快速地生成 C 语言 S-function 代码，这个将在以后的章节中介绍。

C 语言 S-function 与 M 文件 S-function 类似，也是采用回调函数的方式，根据 Simulink 的不同指令调用相应的函数（如初始化、更新离散状态等），然后向 Simulink 返回相应的结果。C 语言 S-function 编译之后产生 MATLAB 可执行文件（MATLAB Executable，MEX），这种可执行文件在不同的操作平台上有不同的方式，对于 Windows 系统，它产生动态链接库（Dynamic-Link Library，DLL）文件。

相对于 M 文件 S-function，C 语言 S-function 拥有更多的回调函数，从而具有更强的处理能力。对于仿真过程中的每一个模块，Simulink 都为之建立一种称为 SimStruct 的数据结构，用于保存该模块的相关信息。而这个结构的内容在 M 文件 S-function 中是不可能读取的，但是在 C 语言 S-function 中，可以读取和修改 SimStruct 结构中的内容，所以，利用 C 语言 S-function，可以实现对仿真过程更强的控制功能。不仅如此，C 语言 S-function 还可以处理更多的数据类型。

C 语言 S-function 的代码一般由 3 部分组成：头部定义、函数体以及尾部定义。在头部定义中，C 语言 S-function 通过宏定义来设置 C 语言 S-function 的名称。函数体部分用来对各种回调函数进行定义。尾部定义则是对 C 语言的 S-function 编译和链接功能的一些参数设置。

（1）C 语言 S-function 模板

前面给出了一个简单的具体示例，下面要详细介绍 C 语言 S-function 模板。Simulink 提出了一个 C 语言 S-function 模板 sfuntmpl_basic.c，该文件存放在“MATLAB 根目录/simulink/src”目录下，它提供了 C 语言 S-function 的基本框架，通过它可以快速地编写函数代码。下面结合 C 语言 S-function 模板代码及其流程图（如图 2-68 和图 2-69 所示）来说明它的构成及具体实现。

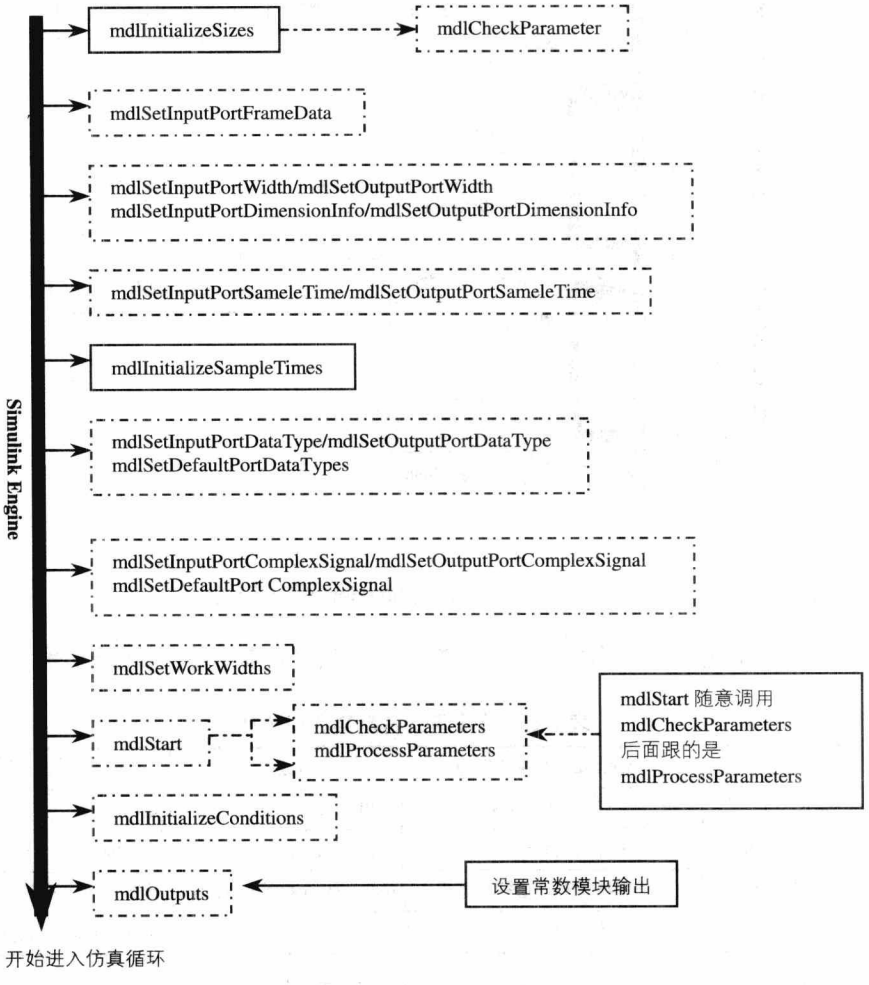


图 2-68　C 语言 S-function 模板初始化流程图

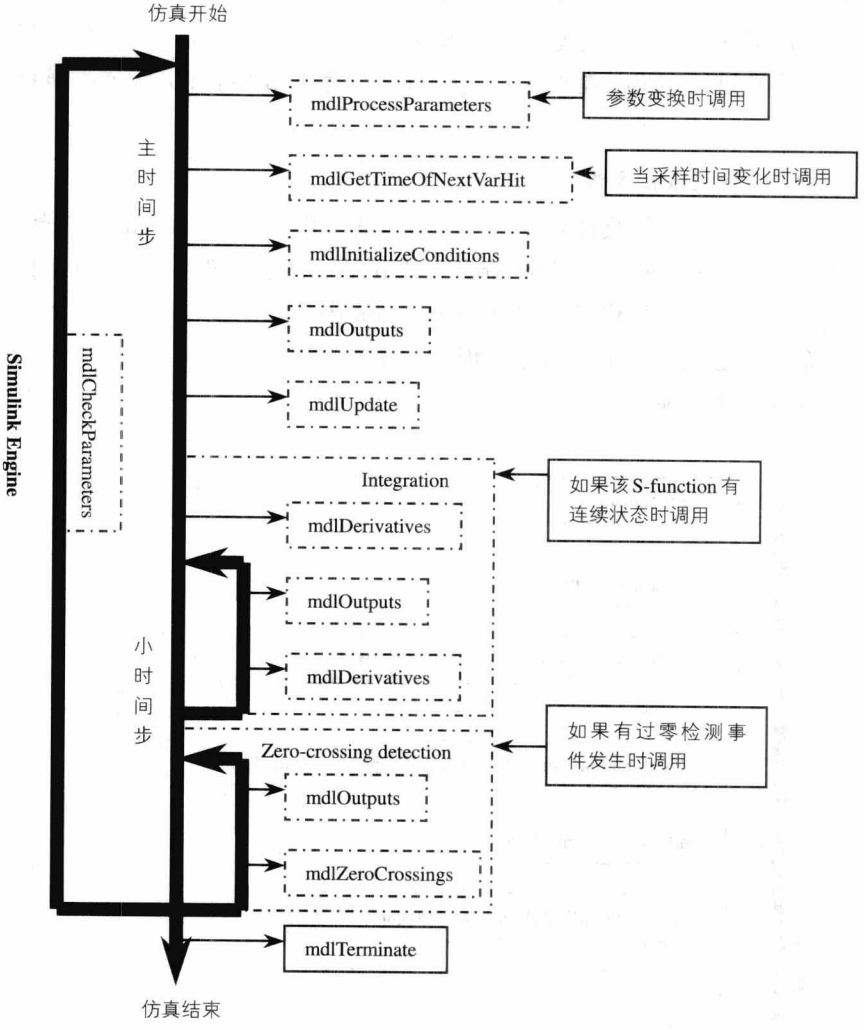


图 2-69　C 语言 S-function 模板流程图

表 2-5 列出了图 2-68 和图 2-69 中一些特殊函数名及其所能实现的功能。

表 2-5　特殊函数名及其实现的功能

函数名	S-function 的对应行为	函数名	S-function 的对应行为
mdlInitializeSizes	使用 ssSet 宏初始化 size 结构	mdlUpdate	更新离散状态
mdlInitializeSampleTimes	初始化采样时间和延迟	mdlOutputs	计算输出
mdlInitializeConditions	设置子系统状态向量初始条件	mdlGetTimeOfNextVarHit	计算下一次的采样时间
mdlDerivatives	计算连续状态微分的值	mdlTerminate	执行必要的结束仿真的任务

接下来给出 C 语言 S-function 模板的程序代码及其注释。

```
/* sfuntmpl_basic.c: C 语言 S-function 模板 */
/* 定义 C 语言 S-function 的名称，用自己的名称替代 sfuntmpl_basic */
#define S_FUNCTION_NAME  sfuntmpl_basic
```

```
/* S-function 的版本，必须设置为 2 */
#define S_FUNCTION_LEVEL 2

/* 文件 simstruc.h 定义了 SimStruct 结构以及相关的宏 */
#include "simstruc.h"

/* S-function 初始化 */
static void mdlInitializeSizes(SimStruct *S)
{
    /* 设置 S-function 的参数个数（用参数数目代替 ssSetNumSFcnParams 中的 0） */
    ssSetNumSFcnParams(S, 0);
    /* 检查 S-function 参数数目是否与设置的一致 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 如果不一致，直接返回 */
        return;
    }

    /* 设置连续状态个数 */
    ssSetNumContStates(S, 0);
    /* 设置离散状态个数 */
    ssSetNumDiscStates(S, 0);

    /* 设置 S-function 输入端口个数为 1 */
    if (!ssSetNumInputPorts(S, 1)) return;
    /* 设置输入端口的宽度 */
    ssSetInputPortWidth(S, 0, 1);
    /* 设置各个输入端口中的元素是否存放在连续内存中 */
    ssSetInputPortRequiredContiguous(S, 0, true);
    /* 设置 S-function 存在直接反馈 */
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    /* 设置 S-function 输出端口数 */
    if (!ssSetNumOutputPorts(S, 1)) return;
    /* 设置 S-function 输出端口宽度 */
    ssSetOutputPortWidth(S, 0, 1);
    /* 设置抽样时间个数（抽样时间 mdlInitializeSampleTines 中定义） */
    ssSetNumSampleTimes(S, 1);
    /* 设置浮点数工作向量的长度 */
    ssSetNumRWork(S, 0);
    /* 设置整数工作向量的长度 */
    ssSetNumIWork(S, 0);
    /* 设置指针工作向量的长度 */
    ssSetNumPWork(S, 0);
    /* 设置工作模式向量的长度 */
    ssSetNumModes(S, 0);
    /* 设置过零点检测状态的个数 */
    ssSetNumNonsampledZCs(S, 0);
    /* 设置 S-function 工作模式的选项 */
    ssSetOptions(S, 0);
}
```

```
/* Function: mdlInitializeSampleTimes */
/* 设置抽样时间 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* 设置抽样时间 */
ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
/* 设置抽样时间偏移 */
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
  /* Function: mdlInitializeConditions */
  /* 初始化连续状态和离散状态 */
  /* 通过 ssGetContState(s)获得连续状态 */
  /* 通过 ssGetRealDisState(s)获得离散状态 */
  static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif
/* MDL_INITIALIZE_CONDITIONS */

#define MDL_START
#if defined(MDL_START)
  /* Function: mdlStart */
  /* 仿真开始时的初始化操作 */
  /* 在整个仿真过程中只执行一次 */
  static void mdlStart(SimStruct *S)
  {
  }
#endif
/*  MDL_START */

/* Function: mdlOutputs */
/* 计算输出信号的数值 */
/* 通过 ssGetY(s)获得输出信号向量 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    /* 获得输入端口的输入信号向量 */
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    /* 获得输出端口的输出信号向量 */
    real_T       *y = ssGetOutputPortSignal(S,0);
    y[0] = u[0];
}

    /* 如果不需要定义 mdlUpdate 函数，则将#define 设置为#undefine */
#define MDL_UPDATE
#if defined(MDL_UPDATE)
  /* Function: mdlUpdate */
```

```
  /* 更新离散状态 */
  /* 每个抽样时刻到来时执行一次 */
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif
/* MDL_UPDATE */

   /* 如果不需要定义 mdlDerivatives 函数，则将#define 设置为#undefine */
#define MDL_DERIVATIVES
#if defined(MDL_DERIVATIVES)
  /* Function: mdlDerivatives */
  /* 计算连续状态的导数 */
  /* 通过 ssGetdX(s)获得连续状态导数向量 */
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif
/* MDL_DERIVATIVES */

/* Function: mdlTerminate */
/* 执行仿真结束时的清理工作 */
static void mdlTerminate(SimStruct *S)
{
}

/* 如果本函数编译成 MEX 文件则链接 simulink.c 文件 */
/* 否则，链接 cg_sfun.h 文件 */
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

注意　以下是常用的几个函数的书写格式，具体详解请参照 MATLAB 帮助文档。

```
ssSetInputPortWidth(SimStruct *S,int_T port,int_T width)
ssSetInputPortRequiredContiguous(SimStruct *S,int_T port,int_T flag)
ssSetInputPortFrameData(SimStruct *S,int_T port,int_T asscptsFrames)
ssSetInputPortComplexSignal(SimStruct *S,input_T port,CSignal_T csig)
ssGetInpupPortSignal(SimStruct *S,inputPortIdx)
ssSetOutputPortWidth(SimStruct *S,int_T port,int_T width)
ssSetSampleTime(SimStruct *S,st_index,time_T period)
ssSetOutputPortFrameData(SimStruct *S,int_T port,int_T asscptsFrames)
ssSetOutputPortComplexSignal(SimStruct *S,output_T port,CSignal_T csig)
ssGetOutpupPortSignal(SimStruct *S,int_T port)
```

（2）C 语言 S-function 示例

在本节里，我们将通过几个示例程序来学习 C 语言 S-function 的设计方法。这些示例

部分取自 MATLAB 自带的程序，内容具有很广泛的代表性，更重要的是这些例子都简单易懂，是 S-function 初学者的经典素材。每个示例程序都详细地介绍了程序的结构和功能，并且列出了程序代码，同时在程序段中还添加了注释，详细介绍了各个代码段的功能。

- 连续状态系统

下面通过一个示例程序来说明连续状态系统的 C 语言 S-function 的编写。这个示例程序是 MATLAB 自带的，位于“MATLAB 根目录/toolbox/simulink/simdemos/ simfeatures”目录下的 sfcndemo_csfunc.mdl 文件。

【示例 2.11】连续状态系统 C 语言 S-function 示例。图 2-70 是该模型的框图。

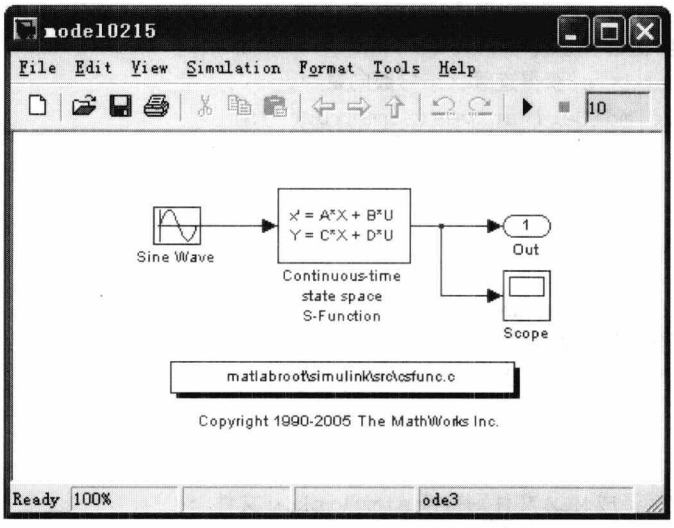


图 2-70　连续状态系统的系统框图

在这个示例程序中，正弦信号产生器产生一个具有两个元素的正弦信号向量，向量中的元素具有连续的抽样时间，信号幅度分别是 1 和 2。Continuous-time state space S-function 模块对输入信号进行变换，然后其输出通过 Scope 模块显示出来。

该连续状态模块由 C 语言 S-function csfunc.c 构造，它位于“MATLAB 根目录/simulink/src”目录下。

该模块对输入信号向量实施矩阵变换，即：

$$\begin{cases} x' = \boldsymbol{A}x + \boldsymbol{B}u \\ y = \boldsymbol{C}x + \boldsymbol{D}u \end{cases} \qquad (式\ 2\text{-}7)$$

其中 $\boldsymbol{A} = \begin{pmatrix} -0.09 & -0.01 \\ 1 & 0 \end{pmatrix}$；$\boldsymbol{B} = \begin{pmatrix} 1 & -7 \\ 0 & -2 \end{pmatrix}$；$\boldsymbol{C} = \begin{pmatrix} 0 & 2 \\ 1 & -5 \end{pmatrix}$；$\boldsymbol{D} = \begin{pmatrix} -3 & 0 \\ 1 & 0 \end{pmatrix}$。

把 C 语言文件经过编译以后产生的动态链接文件和 mdl 文件放置于相同目录下之后便可以启动仿真程序，程序结束后可以得到示波器的输出，如图 2-71 所示。

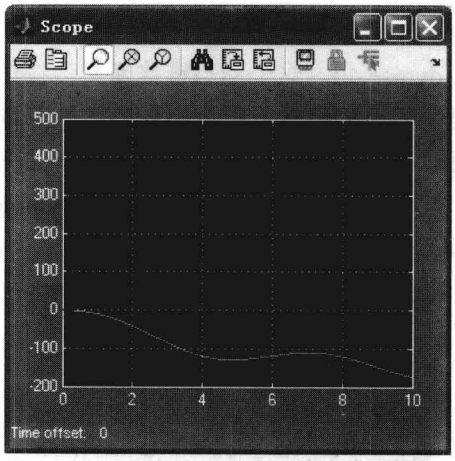


图 2-71　连续状态系统示例程序仿真结果

• 离散状态系统

下面通过一个示例程序来说明离散状态系统的 C 语言 S-function 的编写。这个示例程序是 MATLAB 自带的，位于“MATLAB 根目录/toolbox/simulink/simdemos/ simfeatures”目录下的 sfcndemo_sfun_dynise.mdl 文件。

【示例 2.12】离散状态系统 C 语言 S-function 示例。图 2-72 是该模型的框图。

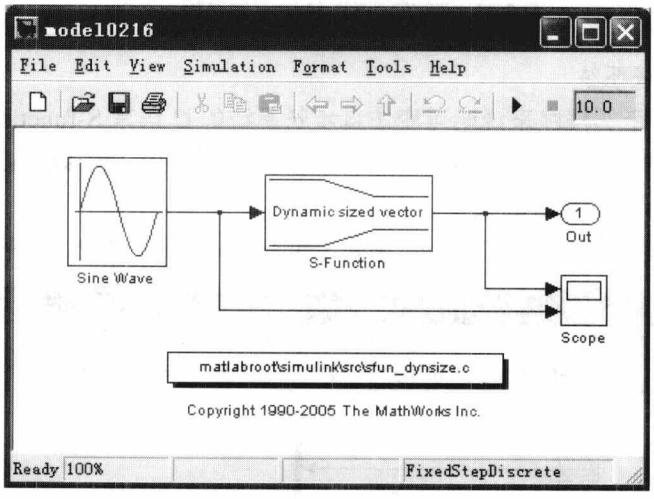


图 2-72　离散状态系统的结构框图

在该仿真模型中，正弦信号发生器产生一个具有 4 个元素的正弦信号向量，向量中的元素都具有连续的抽样时间，信号幅度分别是 1、2、3 和 4。这些信号通过一个 S-function 模块之后产生长度为 2 的输出向量，其中每个信号等于两个输入信号的和，最后由示波器显示仿真结果。

该离散状态模块由 C 语言 S-function 函数 sfun_dynise.c 构造，它位于“MATLAB 根目录/simulink/src”目录下。

最后，在 MATLAB 工作区中输入命令

```
>> mex sfun_dynise.c
```

得到一个 MEX 文件 sfun_dynise.dll。这个动态链接库文件在仿真开始的时候被 Simulink 调入内容，Simulink 将根据仿真进程的需要调用相应的函数获取仿真结果。图 2-73 是仿真结束之后示波器的输出波形，容易看出，正弦信号发生器产生一个 4 个元素的正弦信号向量，信号幅度分别为 1、2、3 和 4。这些信号通过一个 S-function 模块之后产生长度为 2 的输出向量，其中每个信号等于两个输入信号的和。

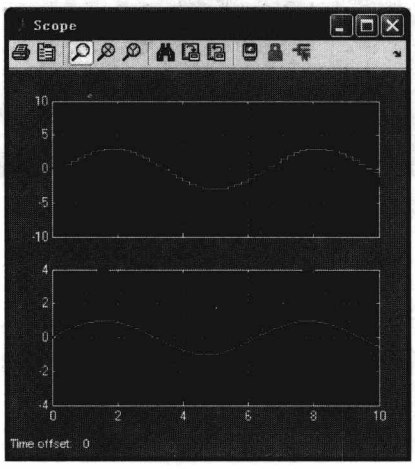


图 2-73　离散仿真系统的示波器输出信号

- 可变仿真步长系统

下面通过一个示例程序来说明可变仿真步长系统的 C 语言 S-function 的编写。这个示例程序是 MATLAB 自带的，位于“MATLAB 根目录/toolbox/simulink/simdemos/ simfeatures”目录下的 sfcndemo_vsfunc.mdl 文件。

【示例 2.13】可变仿真步长系统 C 语言 S-function 示例。图 2-74 是该模型的框图。

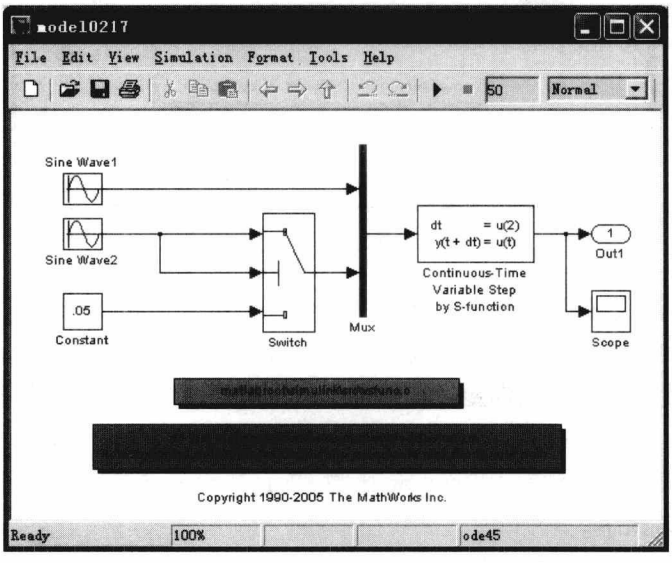


图 2-74　可变仿真步长系统的结构框图

Continuous-Time Variable Step by S-function 的输入信号是由两个信号复用而成的，其中第 1 个输入信号是一个正弦信号，第 2 个信号是一个截掉后的正弦信号，它是由一个幅度为 2 的正弦信号通过一个选通器模块之后得到的。

在此示例中，两个正弦信号发生器分别产生幅度为 1 和 2 的正弦信号，第 2 个正弦信号发生器通过一个选通器模块，该选通器模块有 3 个输入端口，当第 2 个输入端口的输入信号大于一个预先设定的数值时，选通器输出第 1 个输入端口的信号，否则，输出信号等于第 3 个输入端口的输入信号。

第一个正弦信号发生器和选通器模块的输出信号通过 Mux 复合成一个信号进入 Continuous-Time Variable Step by S-function，产生的输出信号通过示波器模块显示。

该模块由 C 语言 S-functionvsfunc.c 构造，它位于“MATLAB 根目录/simulink/src”目录下。

在命令窗口中对该文件进行编译形成动态链接文件，即可开始仿真。图 2-75 是仿真结果，从图中可以看到一些比较尖锐的信号幅度变化，这是由于当第 2 个正弦信号的幅度低于 0.1 时，选通器的输出信号为 0.05，可变步长模块的抽样周期变为 0.05 秒，从而造成比较尖锐的幅度变化。当第 2 个正弦信号的幅度大于 0.1 时，选通器的输出信号就等于这个正弦信号的幅度，这时可变步长模块的抽样周期取决于第 2 个正弦信号的幅度。

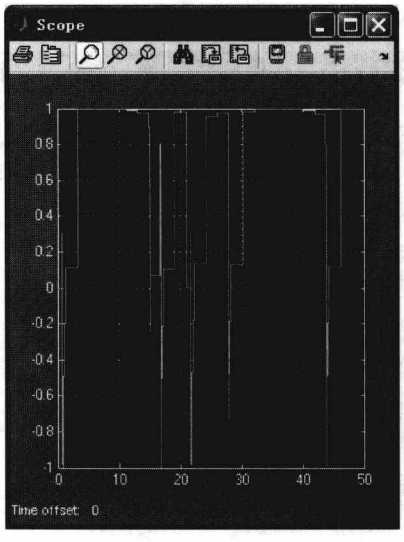


图 2-75　可变仿真步长系统的运行结果

2.4.5　S-function Builder 的使用方法

Simulink 提供了一种自动生成 C 语言 S-function 的工具——S-function Builder，利用它可以编写简单的 C 语言 S-function。S-function Builder 可以根据用户的各种设置自动生成 C 语言 S-function 的框架和代码，通过它用户不需要重新构造函数的框架。但是它在一定程度上限制了 C 语言 S-function 的功能，因为通过 S-function Builder 构造的 S-function 只能有一个输入信号和一个输出信号，而且只能处理 double 类型的数据。

要使用 S-function Builder 创建一个 S-function，应该遵循以下步骤：

- 把 MATLAB 的当前路径设置在想要保存 S-function 的路径上。
- 创建一个新的 Simulink model 文件。
- 从 Simulink 用户自定义函数库复制一个 S-function Builder 模块到新建的模型文件中，如图 2-76 所示。

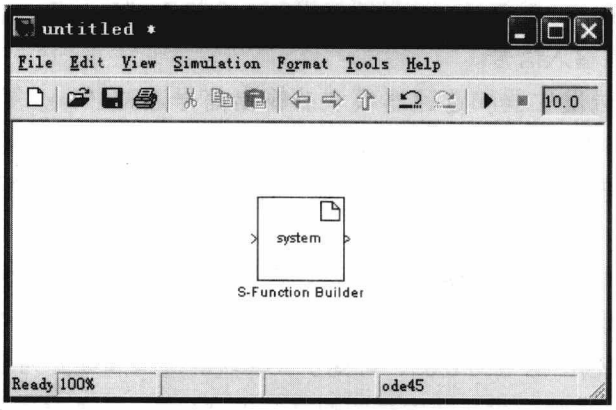


图 2-76　复制 S-function Builder 模块到新建的模型文件中

- 双击这个模块打开 S-function Builder 对话框，如图 2-77 所示。

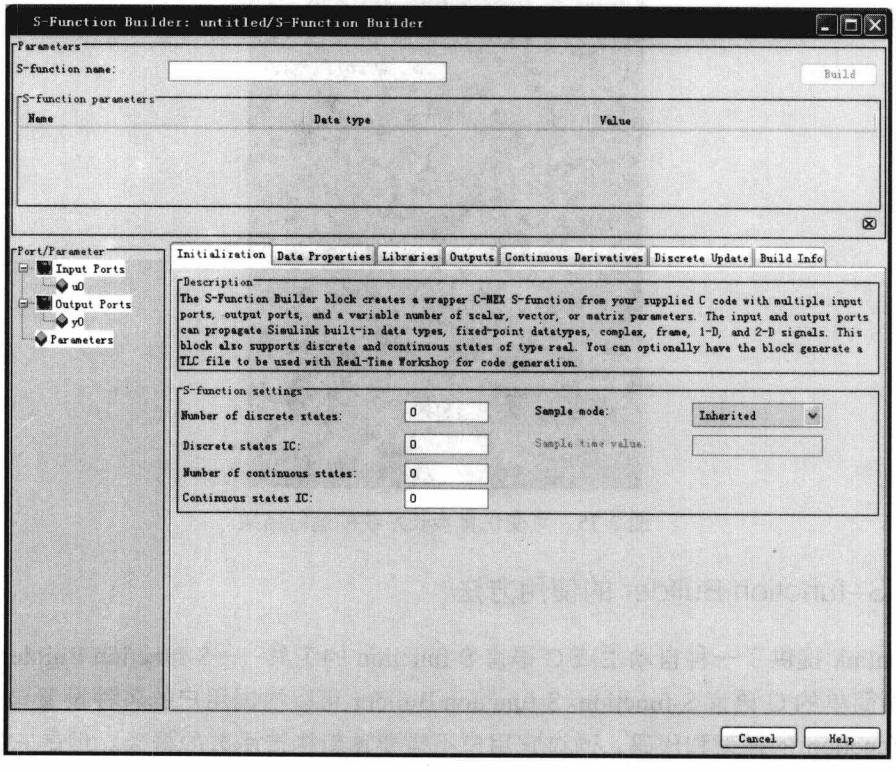


图 2-77　S-function Builder 数据特征界面

- 在 S-function 名字域写入想创建的 S-function 的名字。

- 如果这个 S-function 还有参数，则在 S-function 参数域写入这些参数的默认值。
- 根据 S-function Builder 的书写规范些入一些所需的源程序代码。
- 如果还没有配置 mex 命令，则要在系统中先配置好 MATLAB mex 命令，要配置 mex 命令，在 MATLAB 命令窗口中键入以下命令：

```
>> mex -setup
```

- 单击 S-function Builder 对话框中的 Build 按钮开始创建 S-function 的过程。Simulink 会创建一个执行特殊含义的 S-function MEX 文件并且把它存入当前目录。
- 保存这个包含 S-function 的模型文件。

（1）S-function Builder 创建 S-function 的步骤

S-function Builder 是按照如下的步骤创建 S-funciton 的。首先，它在当前目录下生成下列源文件：

- sfun.c

sfun 是要创建的 S-function 的名字。此文件包含了 C 语言程序源代码，它可以代表生成的 S-function 的标准部分。

- sfun_wrapper.c

此文件包含了读者在 S-function Builder 对话框里输入的传统代码。

- sfun.tlc

此文件使得 Simulink 可以以更快的模式运行创建的 S-function，并且使 RTW 可以把这个 S-function 包含在它产生的代码中。

在产生这个 S-function 源代码以后，S-function Builder 用 MATLAB 的 mex 命令建立 MEX 文件，接着便可以开始使用这个 S-function 了。

（2）设置包含路径

S-function Builder 会通过 MATLAB 的应用数据，即所得的 SfunctionBuilderIncludePath 在指定的路径下寻找头文件。这个数据与用户创建的 S-function Builder 模块的所在模型文件相关联。如果用户的 S-function 使用传统的头文件而这些文件并不在当前目录下，则用户必须更新为 SfunctionBuilderIncludePath 来指定包含这些头文件的目录的位置。SfunctionBuilderIncludePath 是一个三维单元数组，利用它可以指定 3 个包含路径。例如，如下的 MATLAB 命令设置 SfunctionBuilderIncludePath 为两个包含路径：

```
incPath = getappdata(0,'SfunctionBuilderIncludePath');
incPath{1} = '/home/jones/include';
incPath{2} = getenv('PROJECT_INCLUDE_DIR');
setappdata(0,'SfunctionBuilderIncludePath',incPath);
```

（3）S-function Builder 的界面及使用方法

用户可以在 S-function Builder 对话框中输入信息和所需要的传统代码，下面介绍 S-function Builder 对话框及其使用方法。

- 初始化界面

在初始化界面中（如图 2-78 所示），用户可以输入 S-function 的基本特征信息，例如，

输入/输出端口的宽度和抽样时间。

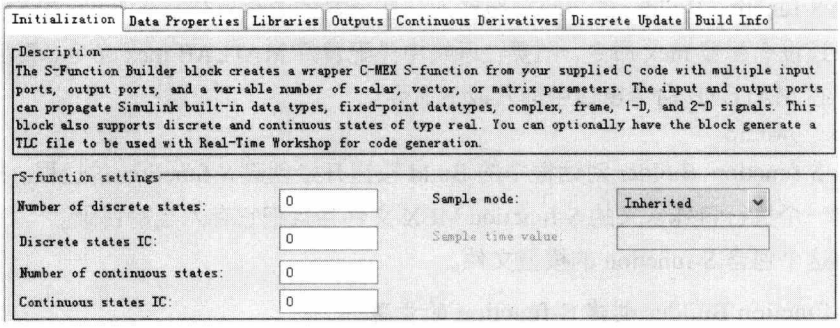


图 2-78　S-function Builder 初始化界面

S-function Builder 利用用户在这个界面的输入信息生成 S-function 的 mdlInitializeSize 调用函数。初始化界面包含以下部分：

- 离散状态数量（Number of discrete states）
- 离散状态初始条件（Discrete states IC）
- 连续状态数量（Number of continuous states）
- 连续状态初始条件（Continuous states IC）
- 抽样模式（Sample mode）
- 抽样时间值（Sample time value）
- 输入端口宽度（Input port width）
- 输出端口宽度（Output port width）
- 参数个数（Number of parameters）

• 数据特征界面

在数据特征界面中，用户可以为自己定义的 S-function 添加参数和端口，如图 2-79 所示。

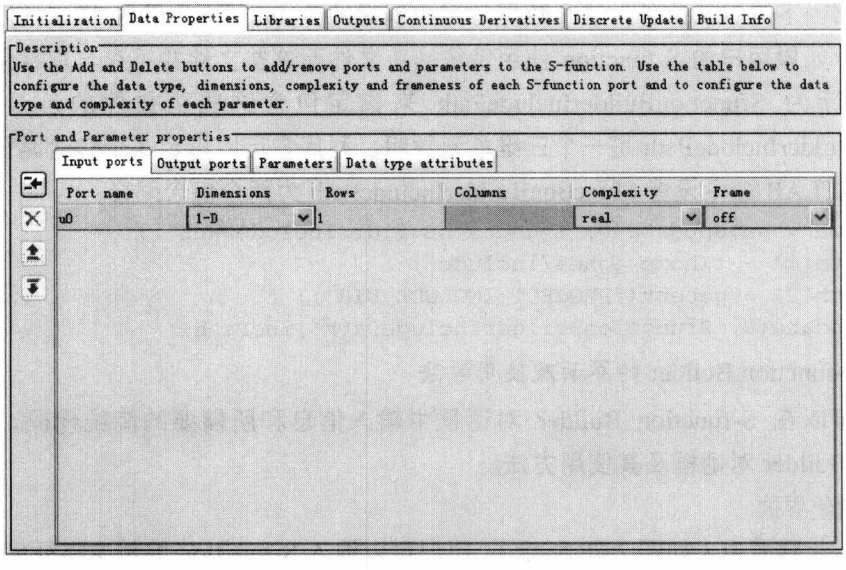


图 2-79　S-function Builder 数据特征界面

在这个界面又分为 3 大部分：输入端口、输出端口和参数。但输入/输出端口部分需要输入的信息是基本一致的：端口名称（Port name）、数据类型（Data type）、端口维数（Dimensions）、行数（Rows）、列数（Columns）、是否为复数信号（Complexity）、数据输入是否为帧格式（Frame）。而参数部分需要输入的是这个参数的名称（Parameters name）、数据类型（Data type）以及是否为复数（Complexity）。

- 输出函数界面

用户可以使用这个界面（如图 2-80 所示）输入 S-funciton 中 outputs 函数的代码。当 S-function Builder 生成 S-function 代码的时候，会将这些代码写在下面程序中的位置（这个程序是 wrapper 函数中的）。

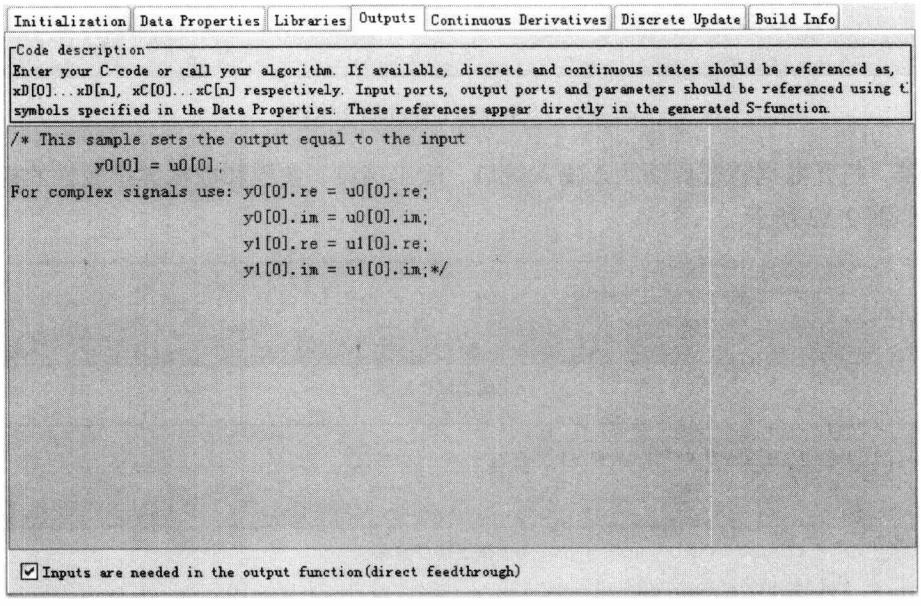


图 2-80　S-function Builder 输出函数界面

```
void sfun_Outputs_wrapper(const real_T *u,
real_T *y,
const real_T *xD,  /* 可选 */
const real_T *xC,  /* 可选 */
cones real_T *param0,  /* 可选 */
int_T p_width0,  /* 可选 */
real_T *param1, /* 可选 */
int_T p_width1,  /* 可选 */
int_T y_width,  /* 可选 */
int_T u_width),  /* 可选 */
{
   /* 用户代码 */
}
```

具体如何编写需要输入的代码，可参看下面的内容。

（4）示例说明

下面向读者介绍一个示例简要说明 S-function Builder 的使用方法。

【示例 2.14】将输入信号放大 x 倍（x 是这个 S-function 的参数）。

首先，创建一个模型文件，将 S-function Builder 拖入其中，双击鼠标左键打开对话框，在 S-function name 后面的文本框中输入 timex（这是要创建的 S-function 的名称），然后在 S-function parameters 的下面输入参数的名字、数据类型以及默认值，如图 2-81 所示。

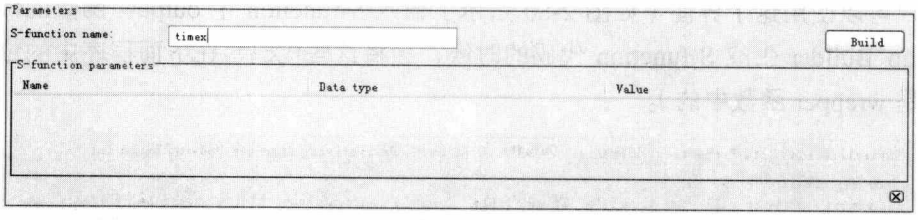
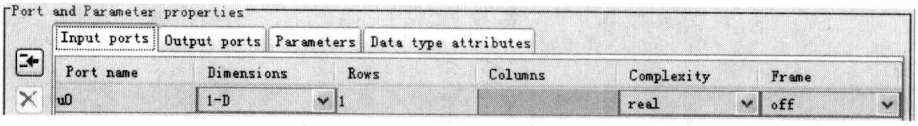
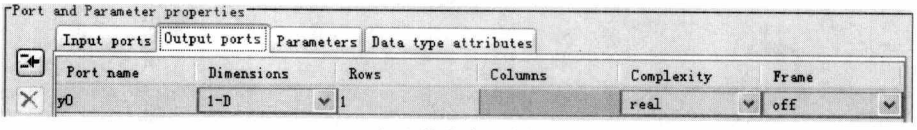


图 2-81　S-function Builder 对话框

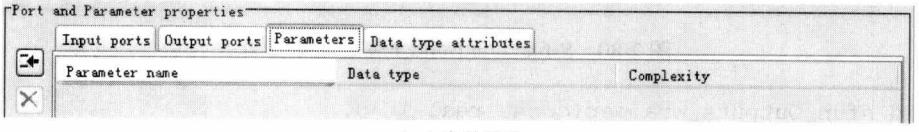
接着，打开数据特征界面写入输入端口、输出端口、参数和数据类型属性的相关特征信息，如图 2-82 所示。



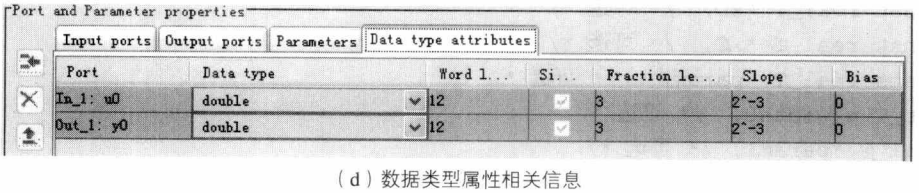
（a）输入端口信息



（b）输出端口信息



（c）参数信息



（d）数据类型属性相关信息

图 2-82　数据特征相关信息

然后，打开 Outputs 输出界面，写入下列代码：

```
real_T times = *TIMES;
*y0 = *u0*times;
```

如图 2-83 所示。

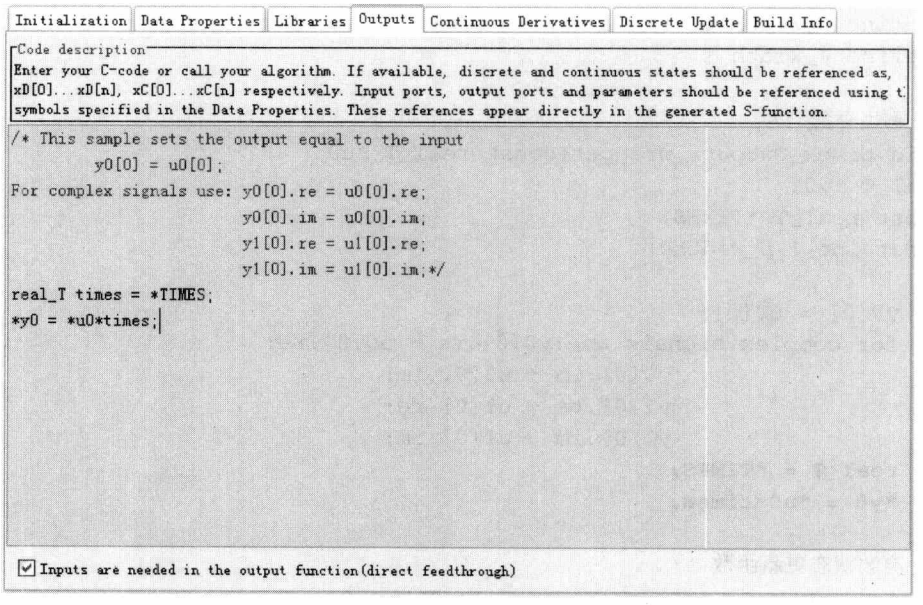


图 2-83　S-function Builder 对话框输出界面

最后，单击 Builder 按钮开始创建过程，完成之后便可以开始仿真。首先建立仿真模型，如图 2-84 所示。

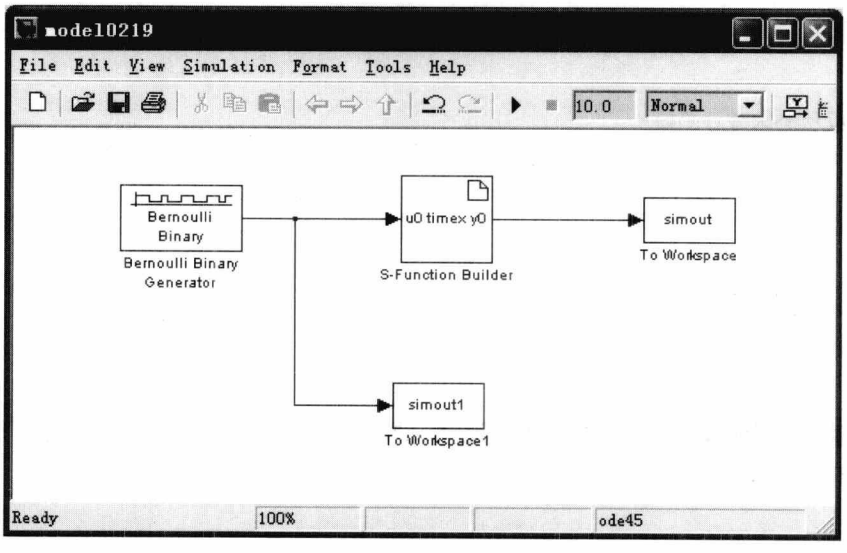


图 2-84　S-function Builder 示例模型

下面给出生成的 timex-wrapper.c 的程序代码，读者通过这个程序代码可以很清楚地了解到自己所写的代码是如何被嵌入的。程序中粗体的语句是在输出界面中写入的代码。

```
/* 所包含的文件 */
#include "tmwtypes.h"
#include <math.h>
```

```
#define u_width 1
#define y_width 1

/* 输出参数 */
void timex_Outputs_wrapper(const real_T *u0,
real_T *y0,
const real_T *TIMES,
const int_T p_width0)
{
    y0[0] = u0[0];
    for complex signals use:y0[0].re = u0[0].re;
                    y0[0].im = u0[0].im;
                    y1[0].re = u1[0].re;
                    y1[0].im = u1[0].im;
    real_T = *TIMES;
    *y0 = *u0*times;
}
/* 离散状态更新函数 */
Void timex_Update_wrapper(const real_T *u0,
const real_T *y0,
const real_T *TIMES,
const int_T p_width0)
{
    /* 代码示例 */
    xD[0] = u0[0];
}

/* 连续状态求微分函数*/
void timex_Derivatives_wrapper(const real_T *u0,
const real_T *y0,
const real_T *TIMES,
const int_T p_width0)
{
    /* 代码示例 */
    dx[0] = xC[0];
}
```

有兴趣的读者可以仔细研究上面的程序，并用以开发其他 S-function 的编写，限于篇幅，在此不再赘述。

2.5 本章小结

本章由浅入深地向读者介绍了 Simulink 的基本操作、子系统及其封装、S-function。重点采用示例的讲解方式，通过大量示例，使读者掌握对 Simulink 模块的基本操作、如何创建子系统并对其进行封装、S-function 的基本概念、如何创建和使用 M 文件和 C 语言 S-function 以及 S-function Builder 的使用方法。熟练掌握本章的内容以后，读者可以根据自己的需要生成相关模块及函数，更好地发挥 MATLAB 的作用。

Part 2

通信系统常用模块仿真篇

第 3 章　信号与信道

通信系统一般由三部分组成，即信源、信道和信宿。信源是通信系统的起点，它产生数据并且对这些数据进行编码和调制，产生适合于信道传输的调制信号；信道是数据信号的传输载体，发送端产生的数据通过信源编码和信号调制转化成调制信号，然后进入信道。这些调制信号通过信道到达接收端，在接收端通过与发送端相反的过程得到原始数据。信宿则是通信系统的终点，它从信道中接收信号，通过解码和解调得到信源端产生的原始数据。

信源、信道和信宿是通信系统中必不可少的三部分。对此，Simulink 通信系统也提供了众多的模块。本章前三节主要介绍信源部分，即信号、序列及噪声的产生模块；第四节介绍几种常见的信道模块；第五节介绍作为信宿的几种常见信号观测设备模块。

3.1　随机数据信号源

在一个通信系统中，信源产生的信号有两种：模拟信号和数字信号。由于现在的各种核心网络都只传输数字信号，因此数字信号成为现代通信系统中不可或缺的信号源。本节将介绍几种数字信号产生器，包括伯努利二进制信号产生器、泊松整数产生器以及随机整数产生器等。

3.1.1　伯努利二进制信号产生器

3.1.1.1　功能与原理

伯努利二进制信号产生器产生符合伯努利分布的随机信号。

伯努利二进制信号产生器产生随机二进制序列，并且在这个二进制序列中的 0 和 1 满足伯努利分布，如公式 3-1 所示。

$$\Pr(x)=\begin{cases}p & x=0\\ 1-p & x=1\end{cases} \qquad (式\ 3\text{-}1)$$

即伯努利二进制信号产生器产生的序列中，产生 0 的概率为 p，产生 1 的概率为 1-p。根据伯努利序列的性质可知，输出信号的均值为为 1-p，方差为 p(1-p)。产生 0 的概率 p 由伯努利二进制信号产生器中的“Probability of a zero”项控制，它可以是 0 和 1 之间的某个实数。

伯努利二进制信号产生器的输出信号，可以是基于帧的矩阵、基于采样的行或列向量，或者基于采样的一维序列。输出信号的性质可以由二进制伯努利序列产生器中的“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”三个选项控制。

3.1.1.2　参数项说明

伯努利二进制信号产生器模块及其参数设定框如图 3-1 所示。

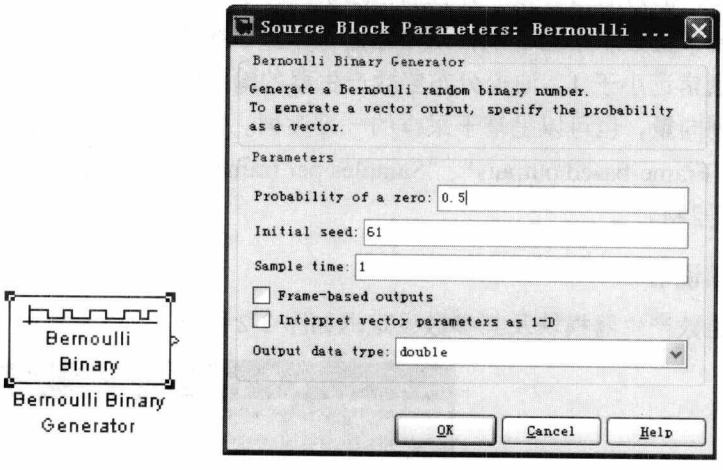


图 3-1　伯努利二进制信号产生器模块及其参数设定框

伯努利二进制信号产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Probability of a zero：伯努利二进制信号产生器输出“0”的几率。对应于公式 3-1 中的 p，为 0 和 1 之间的实数。

Initial seed：伯努利二进制信号产生器的随机数种子，它可以是与“Probability of a zero”项长度相同的矢量或标量。当使用相同的随机数种子时，伯努利二进制信号产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，伯努利二进制信号产生器的输出信号的维数也大于 1。

Sample time：输出序列中每个二进制符号的持续时间。

Frame-based outputs：指定伯努利二进制信号产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Interpret vector parameters as 1-D：如果选中此项，则伯努利二进制信号产生器输出一维序列。否则，输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为 double。

3.1.2　泊松分布整数产生器

3.1.2.1　功能与原理

泊松分布整数产生器产生服从泊松分布的整数序列。

泊松分布整数产生器利用泊松分布产生随机整数。假设 x 是一个服从泊松分布的随机变量，那么 x 等于非负整数 k 的概率可以用公式 3-2 表示。

$$\Pr(k)=\frac{\lambda^k e^{-\lambda}}{k!} \qquad k=0,1,2... \qquad (式\ 3\text{-}2)$$

其中 λ 为一正数，称为泊松参数。并且泊松随机过程的均值和方差都等于 λ。

利用泊松分布整数产生器可以在双传输通道中产生噪声，在这种情况下，泊松参数 λ 应该比 1 小，通常远小于 1。泊松分布整数产生器的输出信号，可以是基于帧的矩阵、基于采样的行或列向量，也可以是基于采样的一维序列。输出信号的性质可以由泊松分布整数产生器中的“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”三个选项控制。

3.1.2.2 参数项说明

泊松分布整数产生器模块及其参数设定框如图 3-2 所示。

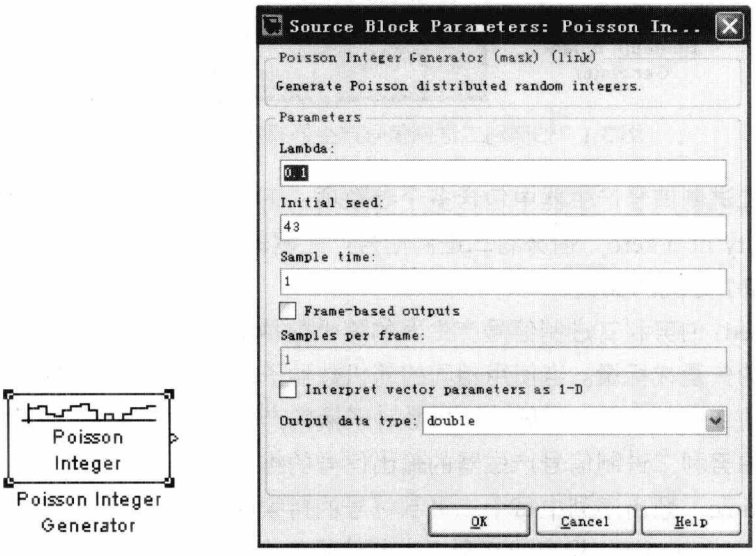


图 3-2 泊松分布整数产生器模块及其参数设定框

泊松分布整数产生器对话框中包含多个参数项，下面分别对各项进行简单的介绍。

Lambda：确定泊松参数 λ，如果输入一个标量，那么输出矢量的每一个元素分享相同的泊松参数。

Initial seed：泊松分布整数产生器的随机数种子。当使用相同的随机数种子时，泊松分布整数产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，泊松分布整数产生器的输出信号的维数也大于 1。

Sample time：输出序列中每个整数的持续时间。

Frame-based outputs：指定泊松分布整数产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则泊松分布整数产生器输出一维序列。否则，输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为 double。

3.1.3　随机整数产生器

3.1.3.1　功能与原理

随机整数产生器的用来产生[0，M-1]范围内具有均匀分布的随机整数。

随机整数产生器输出整数的范围[0，M-1]可以由用户自己设定。M 的大小可在随机整数产生器中的“M-ary number”项中自由输入。M 可以是标量也可以是矢量。如果 M 为标量，那么输出均匀分布且互不相关的随机变量。如果 M 为矢量，其长度必须和随机整数产生器中“Initial seed”的长度相同，在这种情况下，每一个输出对应一个独立的输出范围。如果“Initial seed”是一个常数，那么产生的噪声是周期重复的。

随机整数产生器的输出信号，可以是基于帧的矩阵、基于采样的行或列向量，也可以是基于采样的一维序列。输出信号的性质可以由“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”三个选项控制。

3.1.3.2　参数项说明

随机整数产生器模块及其参数设定框如图 3-3 所示。

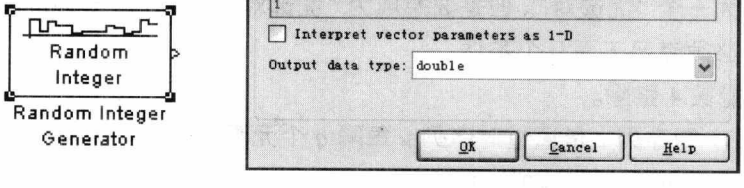


图 3-3　随机整数产生器模块及其参数设定框

随机整数产生器对话框中包含多个参数项，下面分别对各项进行简单的介绍。

M-ary number：输入正整数或正整数矢量，设定随机整数的取值范围。当该参数设置为 M 时，随机整数的取值范围是[0，M-1]。

Initial seed：随机整数产生器的随机数种子。当使用相同的随机数种子时，随机整数产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，随机整数产生器的输出信号的维数也大于 1。

Sample time：输出序列中每个整数的持续时间。

Frame-based outputs：指定随机整数产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则泊松分布整数产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为 double。如果想要输出为 boolean 型，“M-ary number”项必须为 2。

3.2 序列产生器

序列产生器用来产生一个具有某种特性的二进制序列，这种序列可能有比较独特的自相关属性或互相关属性。在码分多址（CDMA）移动通信系统中，通过使用某种序列来达到多址接入效果，如 CDMA 2000 使用了 Walsh 码和 PN 序列，WCDMA 中还使用了 Gold 序列。

MATLAB 中提供了若干种序列产生器，在本节中，我们简单介绍一下其中的 Gold 序列产生器、PN 序列产生器、Walsh 序列产生器等。

3.2.1 Gold 序列产生器

3.2.1.1 功能与原理

Gold 序列产生器用来产生 Gold 序列。Gold 序列的一个重要的特性是其具有良好的互相关性。Gold 序列产生器根据两个长度为 $N=2^n-1$ 的序列 u 和 v 产生一个 Gold 序列 $G(u,v)$，序列 u 和 v 称为一个“优选对”。但是要想成为“优选对”进而产生 Gold 序列，长度 $N=2^n-1$ 的序列 u 和 v 必须满足下面几个条件：

（1）n 不能被 4 整除。

（2）$v=u[q]$，即序列 v 是通过对序列 u 每隔 q 个元素进行一次采样得到的序列，其中 q 是奇数，$q=2^k+1$ 或 $q=2^{2k}-2^k+1$。

（3）n 和 k 的最大公约数满足下面的条件：$\gcd(n,k)=\begin{cases}1, & n\equiv 1 \bmod 2\\ 2, & n\equiv 2 \bmod 4\end{cases}$

由“优选对”序列 u 和 v 产生的 Gold 序列 $G(u,v)$可用公式 3-3 表示：

$$G(u,v)=\{u,v,u\oplus v,u\oplus Tv,u\oplus T^2v,...,u\oplus T^{N-1}v\} \quad (式\ 3\text{-}3)$$

其中 $T^n x$ 表示将序列 x 以循环移位的方式向左移 n 位。⊕代表模二加。需要注意的是由长度为 N 的两个序列 u 和 v 产生的 Gold 序列 $G(u,v)$ 中包含了 N+2 个长度为 N 的序列。Gold 序列产生器可以根据设定的参数输出其中的某一个序列。

如果有两个 Gold 序列 X、Y 属于同一个集合 $G(u,v)$，并且长度 $N=2^n-1$，那么这两个序列的互相关函数只能有三种可能：$-t(n)$、-1、$t(n)-2$。其中

$$t(n)=\begin{cases}1+2^{(n+1)/2} & n \text{ 为偶数} \\ 1+2^{(n+2)/2} & n \text{ 为奇数}\end{cases}$$

Gold 序列实际上是把两个长度相同的 PN 序列产生器产生的“优选对”序列进行异或运算后得到的序列，如图 3-4 所示。

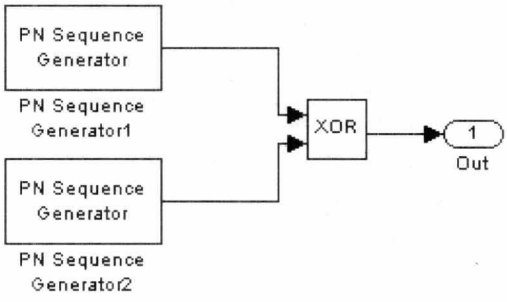


图 3-4　Gold 序列产生器结构示意图

3.2.1.2　参数项说明

Gold 序列产生器模块及其参数设定框如图 3-5 所示。

Gold 序列产生器对话框中包含多个参数项，下面分别对各项进行简单的介绍。

Preferred polynomial[1]：“优选对”序列 1 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

Initial states[1]：“优选对”序列 1 的初始状态。它是一个二进制向量，用于表明与优选对序列 1 对应的 PN 序列产生器中每个寄存器的初始状态。

Preferred polynomial[2]：“优选对”序列 2 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

Initial states[1]：“优选对”序列 2 的初始状态。它是一个二进制向量，用于表明与优选对序列 2 对应的 PN 序列产生器中每个寄存器的初始状态。

Sequence index：用于限定 Gold 序列 $G(u,v)$的输出，其范围是[-2, -1, 0, 1, 2, ..., 2^n–2]，表 3-1 给出了 Sequence index 与序列 $G(u,v)$中元素的对应关系。

Shift：指定 Gold 序列产生器的输出序列的时延。该参数是一个整数，表示序列延迟 Shift 个抽样周期后输出。

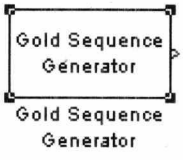
Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定 Gold 序列产生器以帧格式产生输出序列。

Samples per frame：该参数用来确定每帧的抽样点数目。本项只有当“Frame-based outputs”项选中后有效。

Reset on nonzero input：选择该项之后，Gold 序列产生器提供一个输入端口，用于输入复位信号。如果输入不为 0，Gold 序列产生器会将各个寄存器恢复到初始状态。

Output data type：决定模块输出的数据类型，可以是 boolean、double、Smallest、unsigned、integer 等类型，默认为 double。



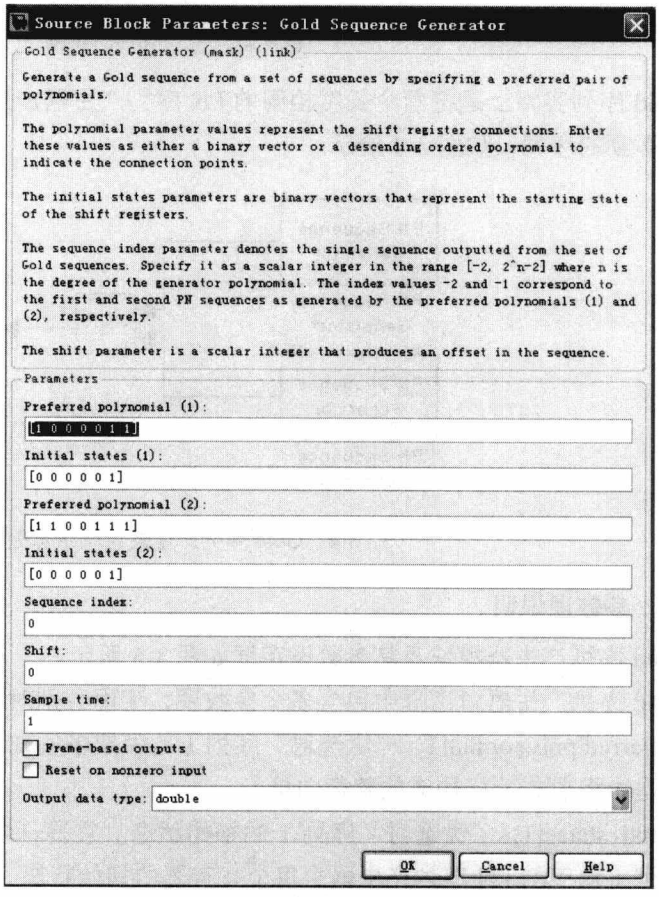


图 3-5　Gold 序列产生器模块及其参数设定框

表 3-1　Sequence index 与序列 $G(u,v)$中元素的对应关系

Sequence Index	序列输出
-2	u
-1	v
0	$u \oplus v$
1	$u \oplus Tv$
2	$u \oplus T^2 v$
…	…
2^n-2	$u \oplus T^{2n-2} v$

3.2.2　PN 序列产生器

3.2.2.1　功能与原理

PN 序列产生器用于产生一个伪随机序列。

PN 序列产生器利用线性反馈移位寄存器（LFSR）来产生 PN 序列，线性反馈移位寄存器可以通过简单的移位暂存器产生器结构得到实现。PN 序列产生器的原理如图 3-6 所示。

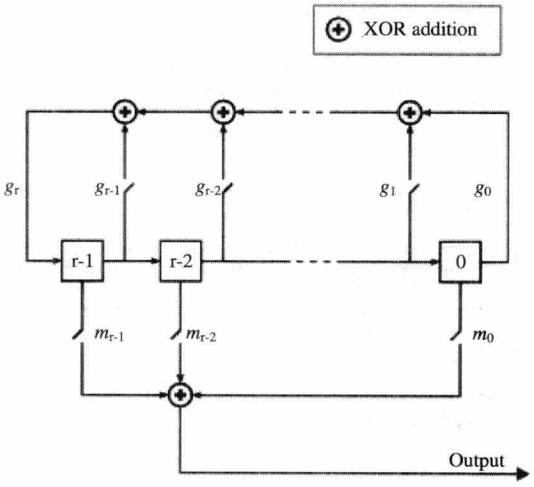


图 3-6　PN 序列产生器原理

由图可知，PN 序列产生器中共有 r 个寄存器，每个寄存器都以相同的抽样频率更新寄存器的状态，即第 k 个寄存器在 t+1 时刻的状态 m_k^{t+1} 等于第 k+1 个寄存器在 t 时刻的状态 m_{k+1}^t。PN 序列产生器可以用一个生成的多项式表示：

$$g_r z^r + g_{r-1} z^{r-1} + g_{r-2} z^{r-2} + \ldots + g_1 z + g_0 \qquad （式 3\text{-}4）$$

式 3-4 中的 g_i 对应图 3-6 中的开关状态：当 g_i 为 1 时，表示开关 g_i 闭合；当 g_i 为 0 时，表示开关 g_i 打开。需要注意的是为了保证生成简单多项式，式 3-4 中的 g_r 和 g_0 必须为 1。

PN 序列产生器产生的多项式有两种表示方式，一种是采用二进制向量的形式，另一种是把生成的多项式中 g_i 不等于 0 的下标 i 组成一个向量。比如生成多项式 $p(z) = z^8 + z^2 + 1$ 可以表示成[1 0 0 0 0 0 1 0 1]，也可以表示成[8 2 0]。

在产生的序列输出之前，PN 序列产生器还可以对由生成多项式产生的序列实行移位或者屏蔽操作，这种操作可以用下面的多项式表示：

$$m_{r-1} z^{r-1} + m_{r-2} z^{r-2} + \ldots + m_1 z + m_0 \qquad （式 3\text{-}5）$$

在式 3-5 中，多项式系数 m_i 对应图 3-6 中的开关 m_i 的状态：当 m_i 为 1 时，表示开关 m_i 闭合；当 m_i 为 0 时，表示开关 m_i 打开。在默认状态下，只有开关 m_0 是关闭的，此时的输出对应于开关 m_0 的寄存器状态。由于多项式 $m_k z^k$ 表示输出对应于开关 m_k 的寄存器状态，它

相对于寄存器 m_0 的时延等于 k，因此，在对输出序列实施移位操作时，只需把式 3-4 对应的参数设置成整数。如果需要对输出序列实施屏蔽操作，则需按照式 3-4 构造多项式。

3.2.2.2 参数项说明

PN 序列产生器模块及其参数设定框如图 3-7 所示。

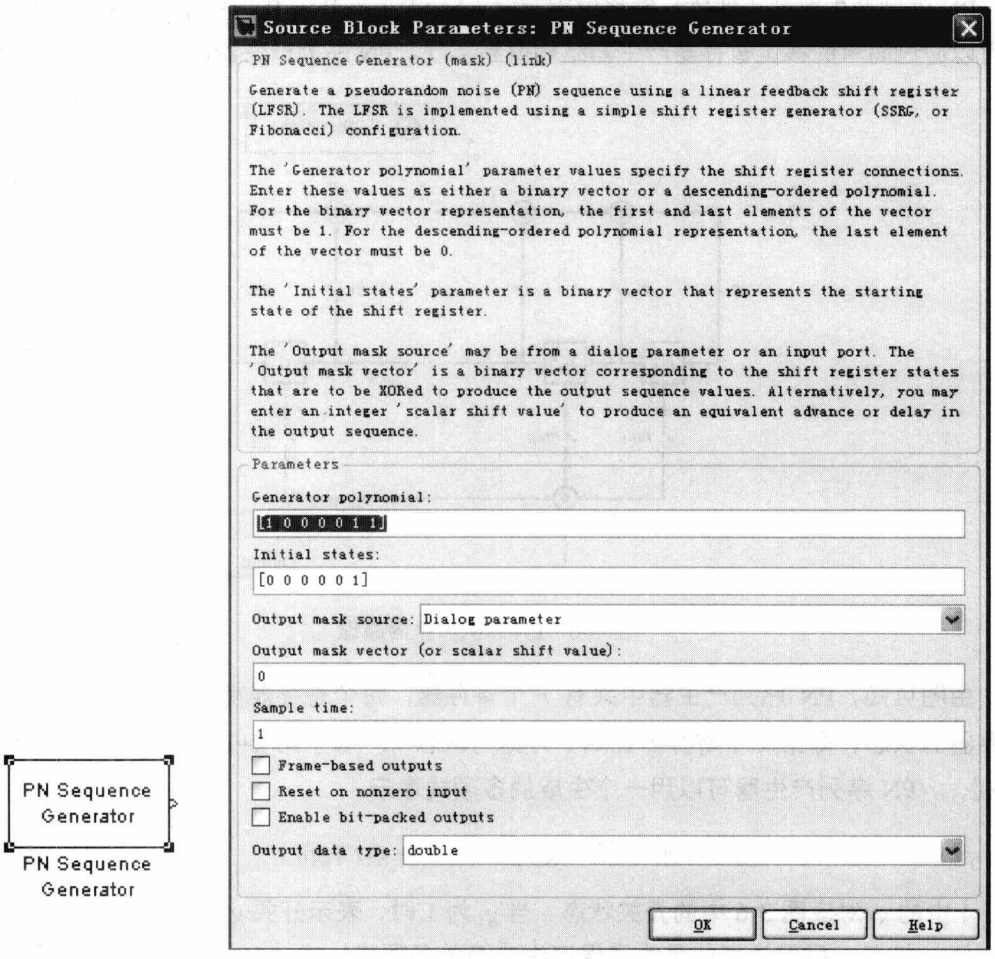


图 3-7　PN 序列产生器模块及其参数设定框

PN 序列产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Generator polynomial：式 3-4 形式的多项式，可以采用二进制向量表示，也可以用由多项式下标构成的整数向量表示。

Initial states：PN 序列产生器中各寄存器的初始状态，表示成二进制向量。

Output mask source：选择模块中的输出屏蔽信息的给定方式。此项为复选框。如果选定 Dialog parameter，则可在“Output mask vector (or scalar shift value)”项中输入；如果选定 Input port，则需要在弹出的对话框中输入。

Output mask vector (or scalar shift value)：给定输出屏蔽（或移位量）。输入的整数或二进制向量决定了生成的 PN 序列相对于初始时刻的延迟。如果移位限定为二进制向量，那

么向量的长度必须和生成多项式的次数相同。此项只有在“Output mask source” 选定为 Dialog parameter 时有效。

Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定 PN 序列产生器以帧格式产生输出序列。

Samples per frame：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”项选中后有效。

Reset on nonzero input：选择该项之后，PN 序列产生器提供一个输入端口，用于输入复位信号。如果输入不为 0，PN 序列产生器会将各个寄存器恢复到初始状态。

Enable bit-packed outputs：选定后激活“Number of packed bits”、“Interpret bit-packed values as signed”两项。

Number of packed bits：设定输出字符的位数（1～32）。

Interpret bit-packed values as signed：有符号整数与无符号整数判断项。如果该项被选定，最高位为 1 时，表示为负。

Output data type：决定模块输出的数据类型，默认为 double。

3.2.3　Walsh 序列产生器

3.2.3.1　功能与原理

Walsh 序列产生器产生一个 Walsh 序列。

如果用 W_i 表示第 i 个长度为 N 的 Walsh 序列，其中 $i=0,1,...,N-1$，并且 Walsh 序列的元素是+1 或-1，$W_i[k]$ 表示 Walsh 序列 W_i 的第 k 个元素，那么对于任意的 i，$W_i[0]=0$。对于任意两个长度为 N 的 Walsh 序列 W_i 和 W_j，有：$W_i W_j^T=\begin{cases}0 & ,i\neq j\\ N & ,i=j\end{cases}$。

长度为 N 的 Walsh 序列实际上是一个 N 阶的 Hadamard 矩阵的一个行向量，因此 Walsh 序列也具有 Hadamard 序列的性质。但 Walsh 序列产生器通过序列中过两点的数目来标识相同长度的各个序列，这跟 Hadamard 序列产生器的序列序号是不同的。对于长度为 N 的 Walsh 序列 W_i，它恰好有 i 个零点。

3.2.3.2　参数项说明

Walsh 序列产生器模块及其参数设定框如图 3-8 所示。

Walsh 序列产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Code length：设定输出 Walsh 序列的长度 N，且需满足 $N=2^n$，$n=0,1,2\cdots$。

Code index：Walsh 序列的序号，为[0，N-1]范围内的整数，表示序列中过零点的数目。

Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定 Walsh 序列产生器以帧格式产生输出序列。

Samples per frame：该参数用来确定每帧的抽样点数目。只有“Frame-based outputs”项选中后，此项才有效。

Output data type：决定模块输出的数据类型，可以是 int8 或 double 等类型，默认为 double。

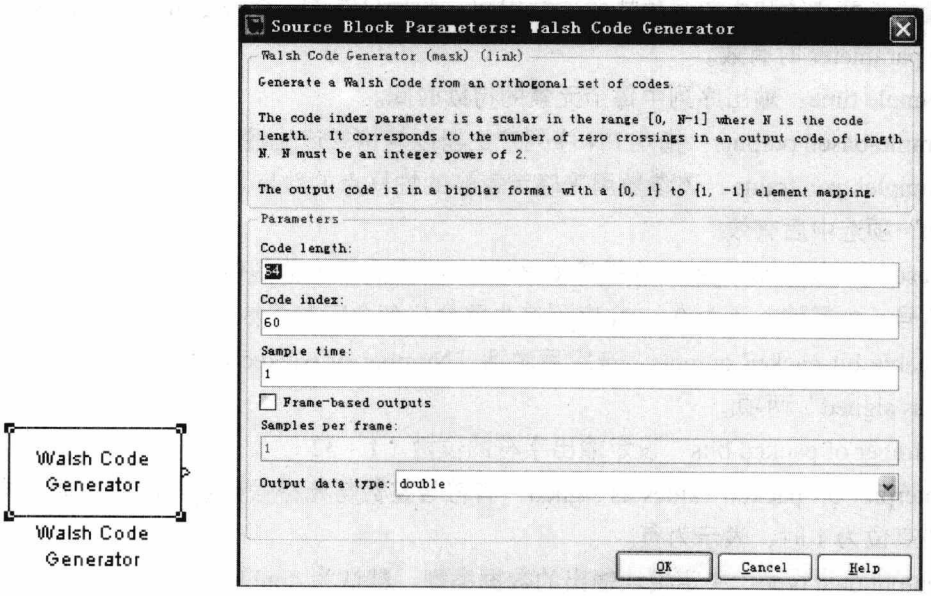


图 3-8　Walsh 序列产生器模块及其参数设定框

3.2.4　其他

除了前面讲到的三种，MATLAB 还另外提供了几种序列产生器，本节对它们做一个简单的介绍。

3.2.4.1　Barker 序列产生器

Barker 序列是 PN 序列的一个子集，具有良好的自相关性。MATLAB 中的 Barker 序列产生器能够产生长度不大于 13 的 Barker 序列。表 3-2 给出了 Barker 序列产生器产生的 Barker 序列。

表 3-2　Barker 序列

码　长	Barker Code
1	[-1]
2	[-1 1]
3	[-1 -1 1]
4	[-1 -1 1 -1]
5	[-1 -1 -1 1 -1]
7	[-1 -1 -1 1 1 -1 1]
11	[-1 -1 -1 1 1 1 -1 1 1 -1 1]
13	[-1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1]

3.2.4.2　Hadamard 序列产生器

Hadamard 序列产生器生成 Hadamard 矩阵中的一个 Hadamard 序列。Hadamard 矩阵中的每一个行向量都是一个 Hadamard 序列，每个行向量之间都是正交的。

Hadamard 矩阵 H_N 是一个 N 行 N 列的方阵，每个元素均为+1 或-1，并且 $N=2^n, n=0,1,2\cdots$。Hadamard 矩阵 $\boldsymbol{H}_N$ 可以通过递归的方式构造，可用下式表示：

$$\boldsymbol{H}_1=[1]$$

$$\boldsymbol{H}_{2N}=\begin{bmatrix}\boldsymbol{H}_N & \boldsymbol{H}_N\\ \boldsymbol{H}_N & -\boldsymbol{H}_N\end{bmatrix}$$

$N\times N$ 阶 Hadamard 矩阵 $\boldsymbol{H}_N$ 的一个重要性质是 $\boldsymbol{H}_N\boldsymbol{H}_N{}^T=N\boldsymbol{I}_N$，其中 $\boldsymbol{I}_N$ 为 N 阶单位矩阵。

3.2.4.3　Kasami 序列产生器

Kasami 序列产生器生成 Kasami 序列。Kasami 序列的长度 $N=2^n-1$（n 为非负偶数），具有良好的互相关性。Kasami 序列可以分成小集合与大集合两种类型。小集合是大集合的子集，但小集合的互相关性更好一些。

假设 u 是一个长度为 N 的二进制序列，那么对 u 每隔 $2^{n/2}+1$ 个元素抽样，得到序列 w，那么 Kasami 序列的小集合可用式 3-6 表示。

$$K_s(u,n,m)=\begin{cases}u & m=-1\\ u\oplus T^m w & m=0,\ldots,2^{n/2}-2\end{cases}\qquad\text{（式 3-6）}$$

$T^m w$ 表示把序列 w 左移 m 位，$\oplus$ 表示模二加。由式 3-6 可以看出，Kasami 序列的小集合共有 $2^{n/2}$ 个序列。

再对序列 u 每隔 $2^{n/2}+1$ 个元素抽样，得到序列 v。当 $n\equiv 2 \bmod 4$ 时，Kasami 序列的大小集合可用式 3-7 表示。

$$K_L(u,n,k,m)=\begin{cases}u & k=-2;m=-1\\ v & k=-1;m=-1\\ u\oplus T^k v & k=0,\cdots,2^n-2;m=-1\\ u\oplus T^m w & k=-2;m=0,\cdots,2^n-2\\ u\oplus T^m w & k=-1;m=0,\cdots,2^n-2\\ u\oplus T^k v\oplus T^m w & k=0,\cdots,2^n-2;m=0,\cdots,2^n-2\end{cases}\qquad\text{（式 3-7）}$$

3.2.4.4　OVSF 序列产生器

OVSF 序列产生器用来生成 OVSF 序列。OVSF 序列最先在 3G 通信系统中引入，用来保持信道间的正交性。

OVSF 序列定义为一个 $N\times N$ 阶矩阵 $\boldsymbol{C}_N$ 的行向量，其中 $N=2^n$。矩阵 $\boldsymbol{C}_N$ 可以采用递归的方法定义，如式 3-8 所示。

另外 OVSF 序列也可用二叉树来表示，具体可见 MATLAB 帮助文件，这里不再赘述。

$$C_1 = [1]$$

$$C_{2N} = \begin{bmatrix} C_N(0) & C_N(0) \\ C_N(0) & -C_N(0) \\ C_N(1) & C_N(1) \\ C_N(1) & -C_N(1) \\ \cdots & \cdots \\ C_N(N-1) & C_N(N-1) \\ C_N(N-1) & -C_N(N-1) \end{bmatrix} \quad (式\ 3\text{-}8)$$

3.3 噪声源发生器

噪声的存在，对通信系统有很大的影响。但噪声本身是一个随机过程，很难通过一种简单的方法预测某个时刻噪声信号的强度。MATLAB 中提供了几种具有不同的随机特征噪声产生器，本节对它们做简单的介绍，包括均匀分布随机噪声产生器、高斯噪声产生器、瑞利噪声产生器、莱斯噪声产生器等。

3.3.1 均匀分布随机噪声产生器

均匀分布随机噪声产生器生成均匀分布的噪声。

均匀分布随机噪声产生器模块的输出数据在指定的最大值与最小值之间，呈现均匀分布。均匀分布随机噪声产生器模块及其参数设定框如图 3-9 所示。

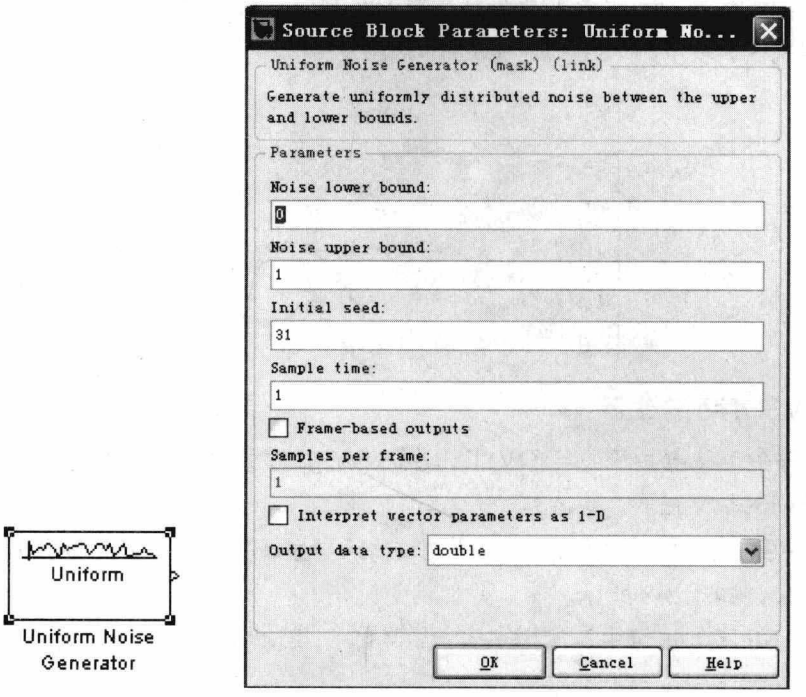


图 3-9 均匀分布随机噪声产生器模块及其参数设定框

均匀分布随机噪声产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Noise lower bound：均匀分布噪声的下界。如果输入为矢量，长度必须和随机数种子（initial seed）相同。

Noise upper bound：均匀分布噪声的上界。如果输入为矢量，长度必须和随机数种子（initial seed）相同。

Initial seed：均匀噪声产生器的随机数种子。当使用相同的随机数种子时，均匀噪声产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。如果随机数种子是一个向量，则均匀噪声产生器的输出信号是一个向量，且它的维数与随机数种子向量维数相同。

Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定均匀噪声产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则均匀噪声产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：模块输出的数据类型设定，有 double 和 single 两种类型。

3.3.2　高斯随机噪声产生器

高斯随机噪声产生器用来产生服从高斯分布的离散噪声信号。

对于 n 维随机变量 $\boldsymbol{X}=(X_1,X_2,...,X_n)$，如果均值为 $u=(u_1,u_2,...,u_n)$，协方差矩阵为 $n\times n$ 阶方阵 $\boldsymbol{K}$，那么 $\boldsymbol{X}$ 等于 x 的概率 $f(x)$ 可由公式 3-9 决定。

$$f(x)=((2\pi)^n \det \boldsymbol{K})^{-1/2} \exp(-(x-u)^T \boldsymbol{K}^{-1}(x-u)/2) \qquad (式\ 3\text{-}9)$$

其中右上角标的 $\boldsymbol{T}$ 表示矩阵的转置；$\det \boldsymbol{K}$ 表示协方差矩阵 $\boldsymbol{K}$ 的行列式的值。

高斯随机噪声产生器模块及其参数设定框如图 3-10 所示。

高斯随机噪声产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Mean value：设定输出的高斯随机变量的均值，可以是标量，也可以是矢量。当输入为标量时，输出的噪声满足一维高斯分布；如果输入为矢量时，输出噪声满足多维高斯分布，且高斯分布的维数为与输入矢量的维数相同。

Variance：设定输出的高斯随机变量的方差或协方差矩阵，可以是标量，也可以是矢量。当其为标量时，产生服从一维高斯分布的噪声；当其为矢量时，输出噪声满足多维高斯分布，且高斯分布的维数为与输入矢量维数相同。在这种情况下，协方差矩阵为对角矩阵，对角元素等于 Variance，非对角元素为 0。此时输出向量的各元素之间互不相关；当输入为 $n\times n$ 阶方阵时，协方差矩阵即为 Variance。

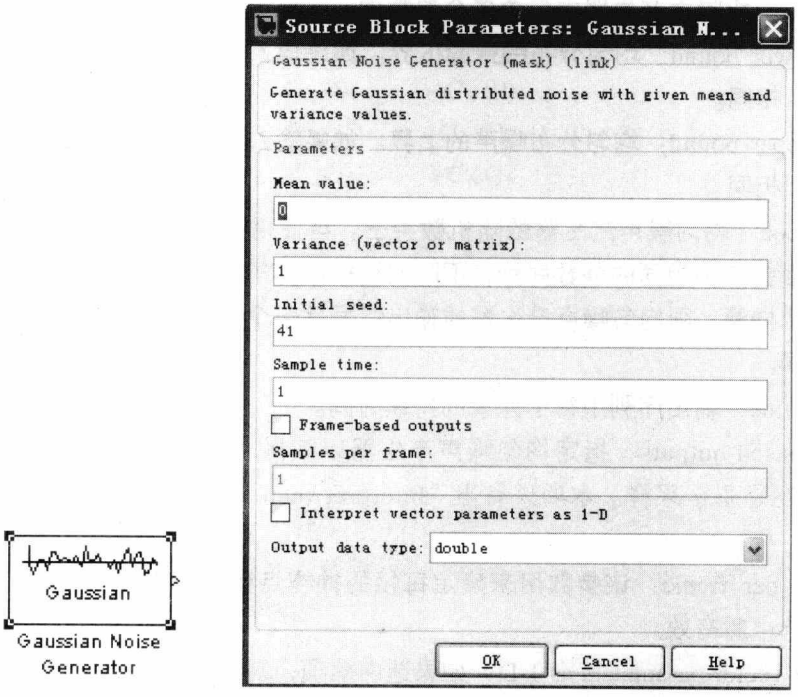


图 3-10　高斯随机噪声产生器模块及其参数设定框

Initial seed：高斯随机噪声产生器的随机数种子。当使用相同的随机数种子时，高斯随机噪声产生器每次产生的整数序列相同，当使用不相同的随机数种子时，每次产生的整数序列也不同。为了获得良好的输出，随机数种子一般输入大于 30 的质数。如果同一模型中还有其他模块需要设定随机数种子参量，最好设定成不同的值。当随机数种子是一 n 维向量时，高斯随机噪声产生器的输出信号服从 n 维高斯分布。

Sample time：输出序列中每个整数的持续时间。

Frame-based outputs：指定高斯随机噪声产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则高斯随机噪声产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：模块输出的数据类型设定，有 double 和 single 两种类型。

3.3.3　瑞利噪声产生器

瑞利噪声产生器用于产生一个服从瑞利分布的随机噪声信号。

根据瑞利分布的定义，如果一个随机变量 x 满足瑞利分布，那么其概率密度函数 $f(x)$ 可由公式 3-10 决定。

$$f(x)=\begin{cases}\dfrac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} & x\geqslant 0\\ 0 & x\leqslant 0\end{cases} \qquad (式\ 3\text{-}10)$$

式中的 σ^2 是决定瑞利分布的参数，称为衰减包络。

瑞利噪声产生器模块及其参数设定框如图 3-11 所示。

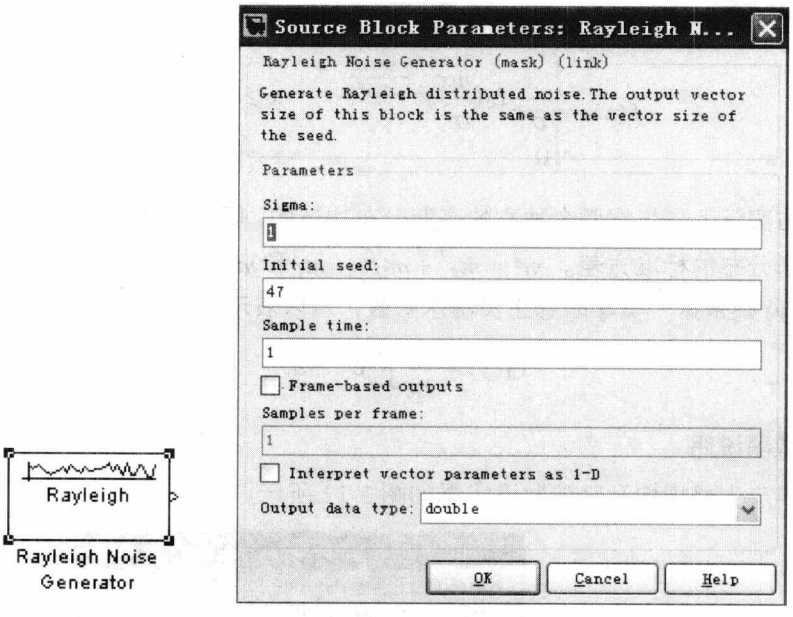


图 3-11　瑞利噪声产生器模块及其参数设定框

瑞利噪声产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Sigma：设定瑞利随机过程的参数，对应于式 3-10 中的参数 σ。

Initial seed：瑞利噪声产生器的随机数种子。当使用相同的随机数种子时，瑞利噪声产生器每次产生的整数序列相同，当使用不相同的随机数种子时，每次产生的整数序列也不同。为了获得良好的输出，随机数种子一般输入大于 30 的质数。如果同一模型中还有其他模块需要设定随机数种子参量，最好设定成不同的值。当随机数种子是一 n 维向量时，瑞利噪声产生器的输出也是一个 n 维向量。

Sample time：输出序列中每个整数的持续时间。

Frame-based outputs：指定高斯随机噪声产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则高斯随机噪声产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：模块输出的数据类型设定，有 double 和 single 两种类型。

3.3.4 莱斯噪声产生器

3.3.4.1 功能与原理

莱斯噪声产生器产生一个服从莱斯分布的随机噪声信号。

根据莱斯分布的定义，如果一个随机变量 x 服从莱斯分布，那么其概率密度函数 $f(x)$ 可由公式 3-11 决定。

$$f(x)=\begin{cases}\dfrac{x}{\sigma^2}I_0\left(\dfrac{mx}{\sigma^2}\right)\mathrm{e}^{-\frac{x^2+m^2}{2\sigma^2}} & x\geqslant 0\\ 0 & x<0\end{cases} \qquad (式\ 3\text{-}11)$$

莱斯分布实际上是根据两个独立的高斯分布构造的，因此在式 3-11 中，σ 为莱斯分布噪声下的高斯分布的标准方差。$m^2=m_I{}^2+m_Q{}^2$，m_I 和 m_Q 分别为两个独立高斯部分的平均值。而 $I_0(y)$ 则是第一类零阶修正贝赛尔函数，可以表示为：

$$I_0(y)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\mathrm{e}^{y\cos t}dt \qquad (式\ 3\text{-}12)$$

3.3.4.2 参数项说明

莱斯噪声产生器模块及其参数设定框如图 3-12 所示。

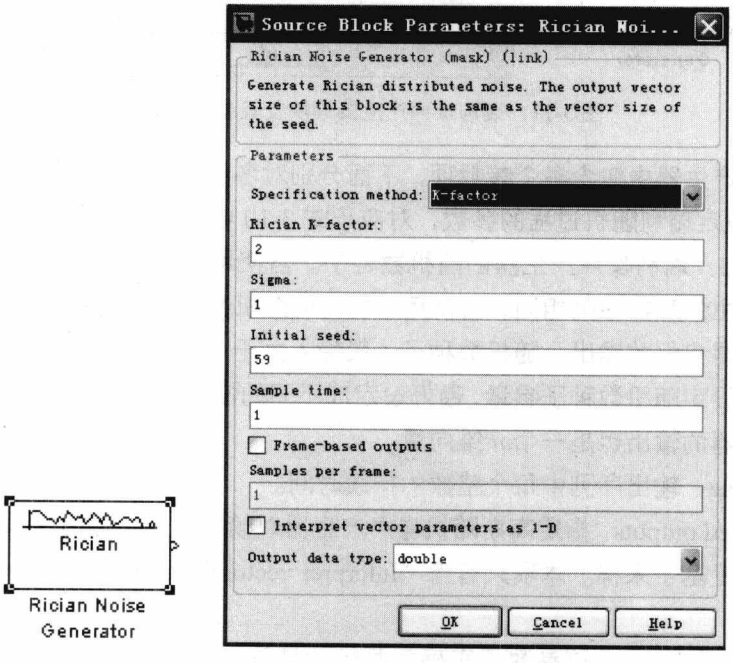


图 3-12　莱斯噪声产生器模块及其参数设定框

莱斯噪声产生器中包含多个参数项，下面分别对各项进行简单的介绍：

Specification method：莱斯分布参数模式复选框，共有两种模式可选，即 K 参数模式或正交分量参数模式。

Rician K-factor：莱斯分布 K 参数设定，$K=m^2/(2\sigma^2)$，m 是式 3-11 中的参数。本项只有当“Specification method”设定为 K-factor 时有效。

In-phase component (mean), Quadrature component (mean)：莱斯分布正交分量均值设定，分别表示两个高斯分布的均值 m_I 和 m_Q。本项只有当“Specification method”设定为 Quadrature components 时有效。

Sigma：标准方差设定，此项即为式 3-11 中的 σ。

Initial seed：莱斯噪声产生器的随机数种子。当使用相同的随机数种子时，莱斯噪声产生器每次产生的整数序列相同，当使用不相同的随机数种子时，每次产生的整数序列也不同。为了获得良好的输出，随机数种子一般输入大于 30 的质数。如果同一模型中还有其他模块需要设定随机数种子参量，最好设定成不同的值。当随机数种子是一 n 维向量时，莱斯噪声产生器的输出也是一个 n 维向量。

Sample time：输出序列中每个整数的持续时间。

Frame-based outputs：指定莱斯噪声产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”项未被选中时有效。

Samples per frame：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”项选中后有效。

Interpret vector parameters as 1-D：如果选中此项，则莱斯噪声产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based outputs”项未被选中时有效。

Output data type：模块输出的数据类型设定，有 double 和 single 两种类型。

3.4 信道

信道是通信系统的基本环节之一。在通信系统中，发送端产生的数据通过信源编码和信号调制转化成调制信号，然后进入信道。这些调制信号通过信道到达接收端，在接收端通过与发送端相反的过程得到原始数据。信道的传输质量影响着信号的接收和解调。

在信号传输的过程中，它会不可避免地受到各种干扰，这些干扰统称为噪声。根据信道中占据主导地位的噪声的特点，信道可以分成加性高斯白噪声信道、多径瑞利退化信道和莱斯退化信道等。本节对它们做简单的介绍。

3.4.1 加性高斯白噪声信道

加性高斯白噪声是最简单的一种噪声，它表现为信号围绕平均值的一种随机波动过程。加性高斯白噪声的均值为 0，方差表现为噪声功率的大小。加性高斯白噪声信道模块的作用就是在输入信号中加入高斯白噪声。加性高斯白噪声信道模块及其参数设定框如图 3-13 所示。

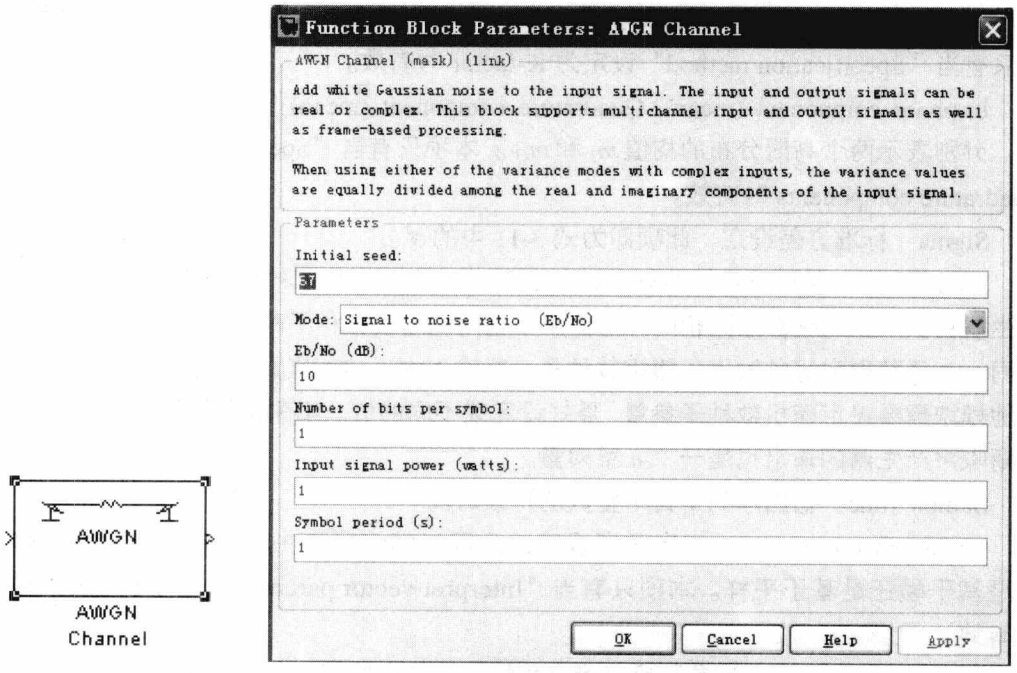


图 3-13　加性高斯白噪声信道模块及其参数设定框

加性高斯白噪声信道模块中包含多个参数项，下面分别对各项进行简单的介绍。

Initial seed：加性高斯白噪声信道模块的初始化种子。不同的初始种子值对应不同的输出，相同的值对应相同的输出。因此具有良好的可重复性，便于多次重复仿真。当输入矩阵为信号时，初始种子值可以是向量，向量中的每个元素对应矩阵的一列。

Mode：加性高斯白噪声信道模块中的模式设定。当设定为 Signal to noise ration（E_b / N_0）时，模块根据信噪比 E_b / N_0 确定高斯噪声功率；当设定为 Signal to noise ration（E_s / N_0）时，模块根据信噪比 E_s / N_0 确定高斯噪声功率，此时需要设定三个参量：信噪比 E_s / N_0、输入信号功率和信号周期。当设定为 Signal to noise ration（*SNR*）时，模块根据信噪比 *SNR* 确定高斯噪声功率，此时需要设定两个参量：信噪比 *SNR* 及信号周期。当设定为 Variance from mask 时，模块根据方差确定高斯噪声功率，这个方差由“Variance”指定，而且必须为正。当设定为 Variance from port 时，模块有两个输入，一个输入信号，另一个输入确定高斯白噪声的方差。

当输入信号为复数形式时，加性高斯白噪声信道模块中的 E_b / N_0、E_s / N_0 和 *SNR* 之间有特定的关系，如式 3-13、式 3-14 所示。

$$E_s / N_0 = (T_{sym} / T_{samp}) \cdot SNR \qquad (式\ 3\text{-}13)$$

$$E_s / N_0 = E_b / N_0 + 10\log_{10}(K) \qquad (式\ 3\text{-}14)$$

在式 3-13 中，T_{sym} 表示输入信号的符号周期，T_{samp} 表示输入信号的抽样周期。式 3-14 中 E_b / N_0 表示比特能量与噪声谱密度的比，K 代表每个字符的比特数。加性高斯白噪声信道模块中复信号的噪声功率谱密度等于 N_0，而在实信号当中，信号噪声的功率谱密度

等于 $N_0/2$，因此对于实信号形式的输入信号，E_s/N_0 和 SNR 之间的关系可以写成式 3-15：

$$E_s/N_0 = 0.5(T_{sym}/T_{samp}) \cdot SNR \qquad (式\ 3\text{-}15)$$

Eb/No (dB)：加性高斯白噪声信道模块的信噪比 E_b/N_0，单位为 dB。本项只有当“Mode”项选定为 Signal to noise ration（E_b/N_0）时有效。

Es/No (dB)：加性高斯白噪声信道模块的信噪比 E_s/N_0，单位为 dB。本项只有当“Mode”项选定为 Signal to noise ration（E_s/N_0）时有效。

SNR (dB)：加性高斯白噪声信道模块的信噪比 SNR，单位为 dB。本项只有当“Mode”项选定为 Signal to noise ration（SNR）时有效。

Number of bits per symbol：加性高斯白噪声信道模块每个输出字符的比特数，本项只有当“Mode”项选定为 Signal to noise ration（E_b/N_0）时有效。

Input signal power (watts)：加性高斯白噪声信道模块输入信号的平均功率，单位为瓦特。本项只有在参数“Mode”设定在 Signal to noise ration（E_b/N_0、E_s/N_0、SNR）三种情况下有效。选定为 Signal to noise ration（E_b/N_0、E_s/N_0）时，表示输入符号的均方根功率；选定为 Signal to noise ration（SNR）时，表示输入抽样信号的均方根功率。

Symbol period (s)：加性高斯白噪声信道模块每个输入符号的周期，单位为秒。本项只有在参数“Mode”设定在 Signal to noise ration（E_b/N_0、E_s/N_0）情况下有效。

Variance：加性高斯白噪声信道模块产生的高斯白噪声信号的方差。本项只有在参数 Mode 设定为 Variance from mask 时有效。

3.4.2　多径瑞利退化信道

3.4.2.1　功能与原理

瑞利退化是移动通信系统中的一种相当重要的退化信道类型，它在很大程度上影响着移动通信系统的质量。在移动通信系统中，发送端和接收端都可能处在不停的运动状态之中，发送端和接收端之间的这种相对运动将产生多普勒频移。多普勒频移与运动速度和方向有关，它的计算公式如式 3-16 所示：

$$f_d = (vf/c)\cos\theta \qquad (式\ 3\text{-}16)$$

其中 v 是发送和接收端之间的相对运动速度，θ 是运动方向和发送端与接收端连线之间的夹角。c 为光速，f 为频率。

多径瑞利退化信道模块实现基带信号多径瑞利退化信道仿真，其输入为标量或帧格式的复信号。它对无限移动通信系统建模有很重要的意义。

3.4.2.2　参数项说明

多径瑞利退化信道模块及其参数设定框如图 3-14 所示。

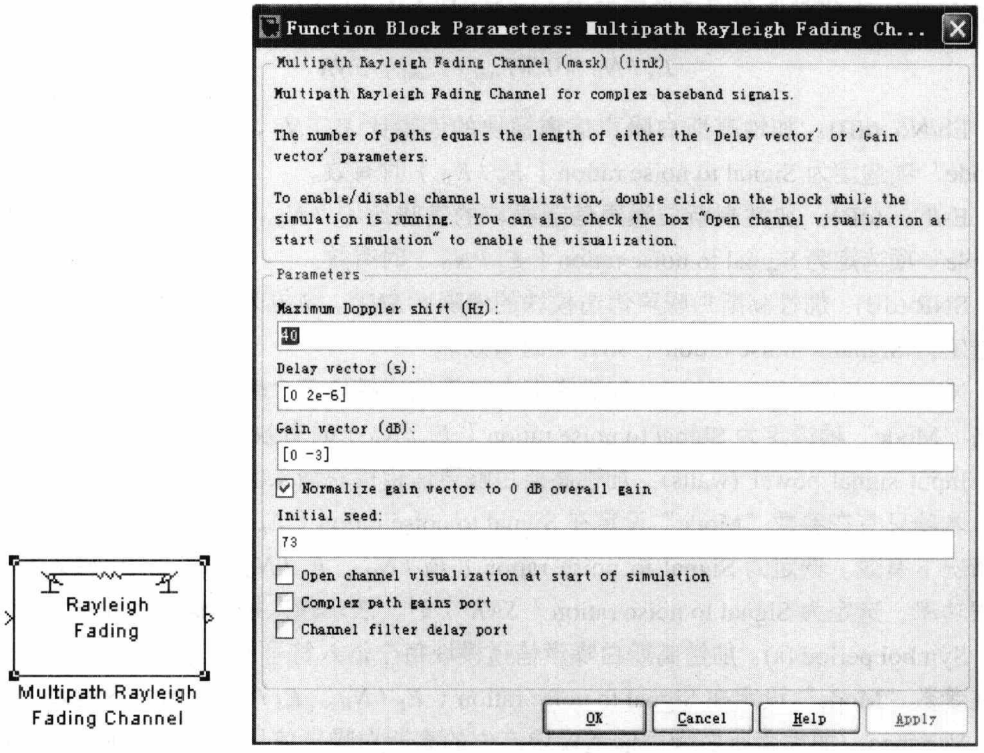


图 3-14　多径瑞利退化信道模块及其参数设定框

多径瑞利退化信道模块中包含多个参数项，下面分别对各项进行简单的介绍。

Maximum Doppler shift (Hz)：多径瑞利退化信道模块的最大多普勒频移，单位为 Hz。

Delay vector (s)：多径瑞利退化信道模块输入信号各路径的时延，单位为秒。

Gain vector (dB)：多径瑞利退化信道模块输入信号各路径的增益，单位为 dB。

Normalize gain vector to 0 dB overall gain：选定本参数后，多径瑞利退化信道模块把参数 Gain vector 乘上一个系数作为增益向量，使得所有路径的接收信号强度和等于 0dB。

Initial seed：多径瑞利退化信道模块的初始化种子。

Open channel visualization at start of simulation：多径瑞利退化信道模块中通道可视化选项。选定该项，仿真开始时将会打开通道可视化工具。

Complex path gains port：多径瑞利退化信道模块复数路径增益端口项。选定后，输出每个通道的复数路径增益。这是一个 $N \times M$ 多通道结构，其中 N 为每帧样品数，M 为离散的路径数。

Channel filter delay port：多径瑞利退化信道模块信道滤波延迟端口项，选定后，输出本模块中由于滤波引起的延迟。单路径时，延迟为 0；多路径时，延迟大于 0。

3.4.3　多径莱斯退化信道

在移动通信系统中，如果发送端和接收端之间存在着一条占优势的视距传播路径，这种信号就可以模拟成多径莱斯退化信道。当发送端和接收端之间既存在着视距传播路径，

又有多条反射路径时。它们之间的信道可以同时用多径莱斯退化信道模块和多径瑞利退化信道来进行仿真。

多径莱斯退化信道模块对基带信号的多径莱斯退化信道进行仿真，其输入为标量或帧格式的复信号。多径莱斯退化信道模块及其参数设定框如图 3-15 所示。

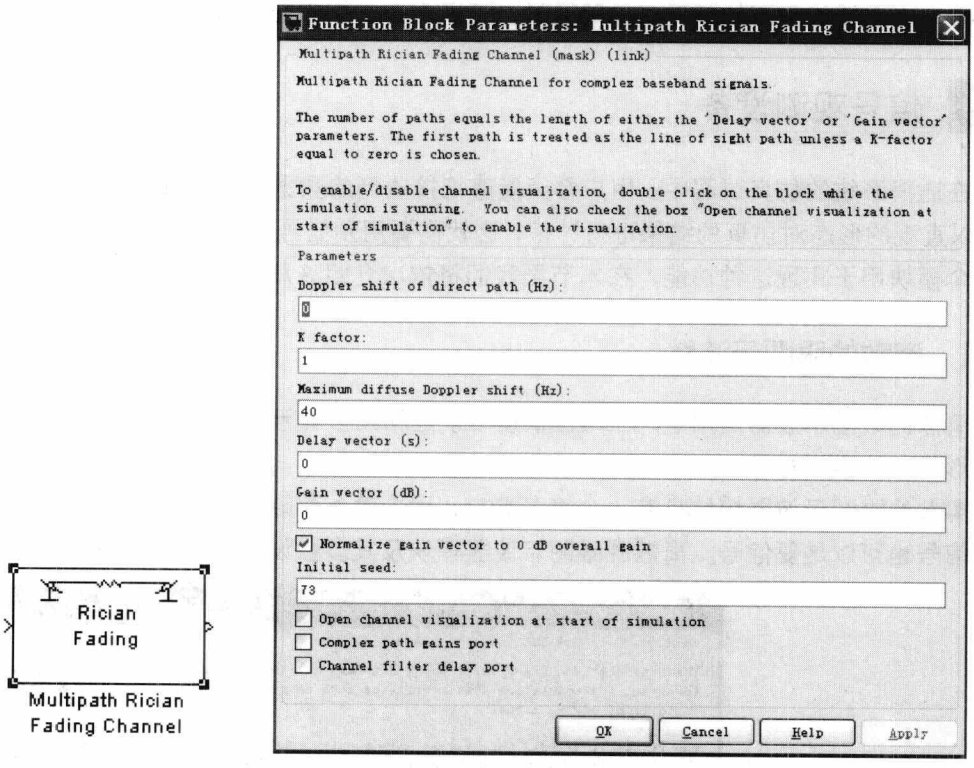


图 3-15　多径莱斯退化信道模块及其参数设定框

多径莱斯退化信道模块中包含多个参数项，下面分别对各项进行简单的介绍。

Doppler shift of direct path (Hz)：多径莱斯退化信道模块中的视距传播路径多普勒频移，单位为 Hz。

K-factor：多径莱斯退化信道模块中的 K 因子。它表示视距传播路径的能量与其他多径信号的能量之间的比值。K 因子越大，表示发送端和接收端之间的视距传播路径的能量越强；当 K 因子等于 0 时，发送端和接收端之间不存在视距传播路径，此时莱斯退化信道就演变成瑞利退化信道。

Maximum diffuse Doppler shift (Hz)：多径莱斯退化信道模块中最大的扩散多普勒频移设定，必须为正数。

Delay vector(s)：多径莱斯退化信道模块中每个路径的传输延迟向量设定。

Gain vector (dB)：多径莱斯退化信道模块中每个路径的增益向量设定。

Initial seed：多径莱斯退化信道模块的初始化种子。

Open channel visualization at start of simulation：多径莱斯退化信道模块中通道可视化选项。选定该项，仿真开始时将会打开通道可视化工具。

Complex path gains port：多径莱斯退化信道模块复数路径增益端口项。选定后，输出每个通道的复数路径增益。这是一个 $N \times M$ 多通道结构，其中 N 为每帧样品数，M 为离散的路径数。

Channel filter delay port：多径莱斯退化信道模块信道滤波延迟端口项。选定后，输出本模块中由于滤波引起的延迟。单路径时，延迟为 0；多路径时，延迟大于 0。

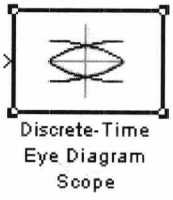
3.5 信号观测设备

在通信系统的仿真过程中，用户希望能够把接收到的数据通过某种方式保存或显示出来，以直观的形态对仿真的结果进行评估，这就需要用到信号观测设备。MATLAB 提供了若干个模块用于实现这种功能，在本节里我们将依次介绍这几种模块。

3.5.1 离散的眼图示波器

离散的眼图示波器是用于产生眼图的模块。通常用于显示调制信号的输出特性，比如脉冲形状、信道失真等。

离散的眼图示波器模块只有一个输入端口，用于输入离散的时间信号。这个信号可以是实信号也可以是复信号。离散的眼图示波器模块及其参数设定框如图 3-16 所示。



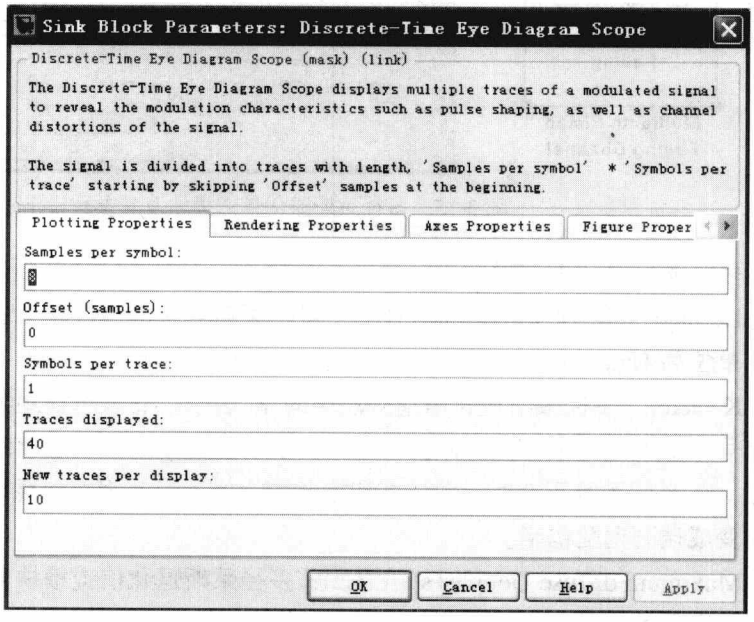


图 3-16 离散的眼图示波器模块及其参数设定框

由图 3-16 可见，离散的眼图示波器参数设定框中又包含四个选项，这是因为离散的眼图需要设定的参数较多，为了方便起见，MATLAB 将其归结为四类，分别为“Plotting Properties”、“Rendering Properties”、“Axes Properties”、“Figure Properties”，默认项为“Plotting Properties”。选定某一类后，参数设定框相应变化。下面分别就这四类做简单的介绍。

3.5.1.1　Plotting Properties 类参数项说明

Plotting Properties：用来设定眼图的绘制方式。该项为默认项，如图 3-16 所示。其中包含如下参数：

Samples per symbol：设定每个符号的抽样数。和“Symbols per trace”项共同决定每径的抽样数。

Offset (samples)：开始绘制眼图之前应该忽略的抽样点的个数。该项一定要是小于“Samples per symbol”和“Symbols per trace”项的非负整数。

Symbols per trace：对于每一个输入信号，离散的眼图示波器模块可以同时绘制多条曲线，每条曲线称为一个径，它们在时间上相差一定的时间周期。本项用来设定每径上的抽样周期。

Traces displayed：设定模块中显示的径的数目，应该为一正整数。

New traces per display：每次显示需要重新绘制的径的数目，该项应是比“Traces displayed”小的正整数。

3.5.1.2　Rendering Properties 类参数项说明

“Rendering Properties”用来设定绘图属性。选定该项后，显示如图 3-17 所示的参数设定框。

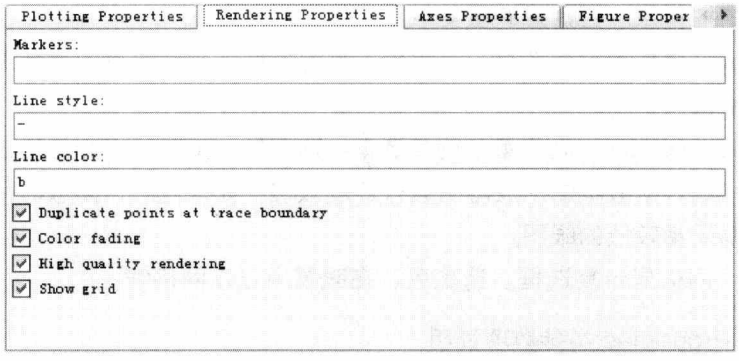


图 3-17　“Rendering Properties”参数设定框

由图 3-17 可知，该项包含下面几个参数：

Markers：设定眼图中每个抽样点的绘制方式。本模块提供了多种方式，如当设置为字符“+”时，则在每个抽样点上显示一个加号。具体如表 3-3 所示。

表 3-3　参数 Marker 的对应关系

Marker	说　明	形　状
Plus	每个抽样点上显示一个加号	
Circle	每个抽样点上显示一个圆圈	
Asterisk	每个抽样点上显示一个乘号	
Point	每个抽样点上显示一个点	
Cross	每个抽样点上显示一个叉号	

Line style：设定眼图中的线型，本模块中共提供了四种线型，如表 3-4 所示。

表 3-4　眼图中线型

Line style	形　状
Solid	———————————
Dashed	— — — — — — — —
Dotted	------------------------------
Dash-dot	— · — · — · — · — · — · — · —

Line color：设定眼图中线条的颜色，本模块中共提供了五种线条颜色，如表 3-5 所示。

表 3-5　眼图中线条颜色

Color	RGB 值	颜　色
Black	(0,0,0)	黑色
Blue	(0,0,255)	蓝色
Red	(255,0,0)	红色
Green	(0,255,0)	绿色
Dark purple	(192,0,192)	暗紫色

Duplicate points at trace boundary：迹边界重复点为复选框，选定后则在每条迹的边缘出现一个重复的抽样点。否则没有重复点。

Color fading：颜色渐变复选框。选定后，眼图中每条迹上的点的颜色深度随着仿真时间的推移而逐渐减弱。

High quality rendering：高质量绘图复选框。选定后，离散时间眼图通过光栅操作绘制高质量的眼图，此时离散眼图示波器的运行速度较慢；如未选定，离散时间眼图通过异或操作快速地绘制质量较低的眼图。

Show grid：网格显示复选框。选定后，在眼图中显示网格线。

3.5.1.3　Axes Properties 类参数项说明

“Axes Properties”用来设定眼图中的坐标轴属性。选定该项后，显示如图 3-18 所示的参数设定框。

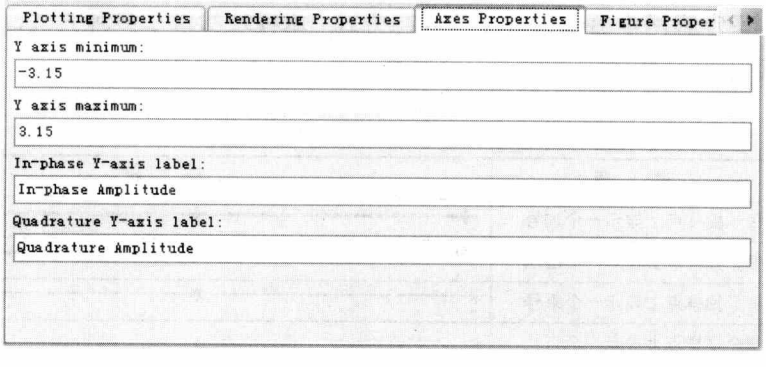


图 3-18　“Axes Properties”参数设定框

由图 3-18 可知，该项包含下面几个参数：

Y-axis minimum：设定纵坐标（即输入信号强度）的最小值。

Y-axis maximum：设定纵坐标（即输入信号强度）的最大值。

In-phase Y-axis label：设定是否显示与 I 支路输入信号对应的纵坐标的标签。

Quadrature Y-axis label：设定是否显示与 Q 支路输入信号对应的纵坐标的标签。

3.5.1.4　Figure Properties 类参数项说明

“Figure Properties”用来设定眼图属性。选定该项后，显示如图 3-19 所示的参数设定框。

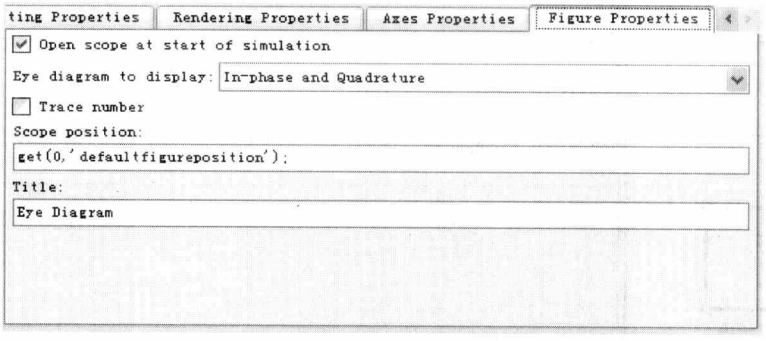


图 3-19　“Figure Properties”参数设定框

由图 3-19 可知，该项包含下面几个参数：

Open scope at start of simulation：本项为复选框。选定后，眼图将在仿真开始的时候自动打，否则，用户需要双击离散眼图示波器之后才能显示。

Eye diagram to display：确定眼图中显示的那个支路的输入信号。当选择 In-phase and Quadrature 时，同时显示实信号和复信号；当选择 In-phase Only 时，只显示实信号。

Trace number：设定是否在眼图中显示当前正在绘制的迹的编号。

Scope position：设定眼图的位置。它是由[left bottom width height]四个元素组成的向量。left 和 bottom 分别表示眼图左下角的纵坐标和横坐标，如(0,0)表示眼图的左下角。width 和 height 分别表示眼图的宽度和高度。

Title：设定眼图的标题。

3.5.2　星座图观测仪

星座图观测仪又称离散时间发散图观测仪，通常用来观测调制信号的特性和信道对调制信号的干扰特性。星座图观测仪模块接收复信号，并且根据输入信号绘制发散图。星座图观测仪模块只有一个输入端口，输入信号必须为复信号。星座图观测仪模块及其参数设定框如图 3-20 所示。

由图 3-20 可见，星座图观测仪参数设定框中又包含四个选项，这是因为星座图需要设定的参数较多，为了方便起见，MATLAB 将其归结为四类，分别为“Plotting Properties”、

“Rendering Properties”、“Axes Properties”、“Figure Properties”，默认项为“Plotting Properties”。选定某一类后，参数设定框相应变化。下面分别就这四类做简单的介绍。

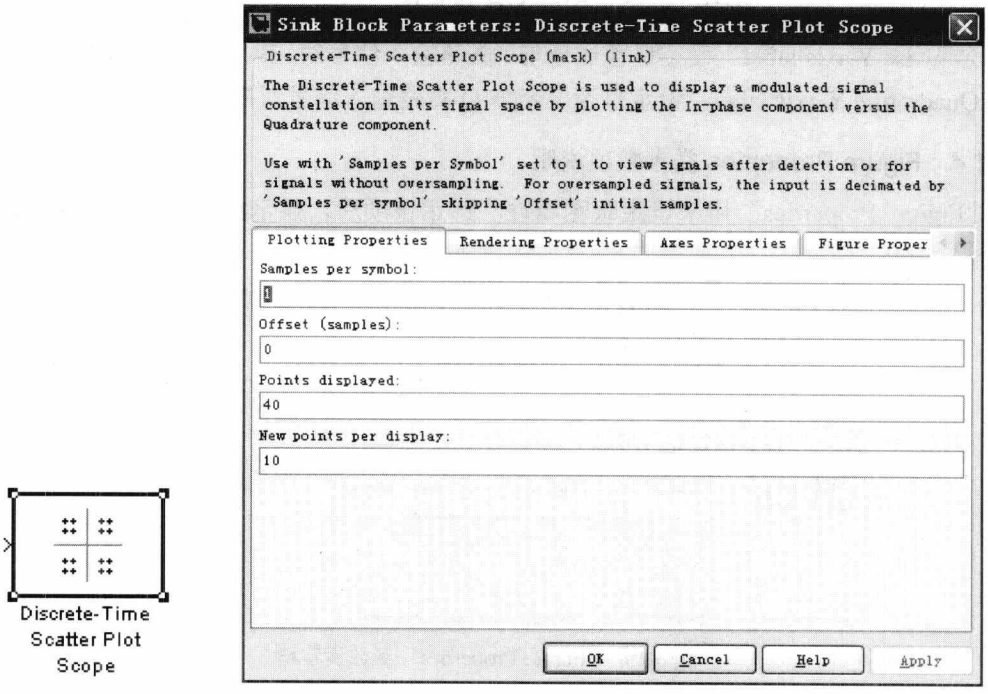


图 3-20　星座图观测仪模块及其参数设定框

3.5.2.1　Plotting Properties 类参数项说明

“Plotting Properties”用来设定星座图的绘制方式。该项为默认项，如图 3-20 所示。其中包含如下参数：

Samples per symbol：设定星座图中每个符号的抽样点数目。

Offset (samples)：开始绘制星座图之前应该忽略的抽样点个数。该项一定要是小于“Samples per symbol”项的非负整数。

Points displayed：星座图观测仪模块中显示的点数目。

New points per display：设定星座图观测仪模块中每次需要重新设定的点数目。

3.5.2.2　Rendering Properties 类参数项说明

“Rendering Properties”用来设定绘图属性。选定该项后，显示如图 3-21 所示的参数设定框。

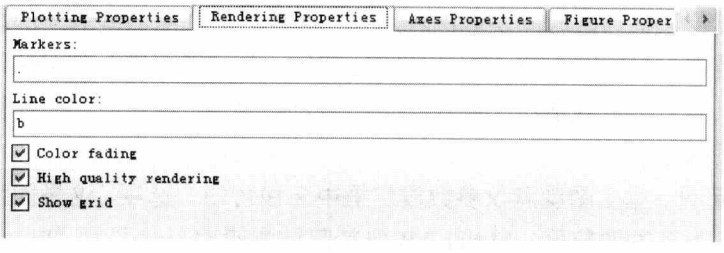


图 3-21　“Rendering Properties”参数设定框

由图 3-21 可知，该项包含下面几个参数：

Markers：设定星座图中每个抽样点的绘制方式，同表 3-3。

Line color：设定眼图中线条颜色，同表 3-5。

Color fading：颜色渐变复选框。选定后，星座图中每条迹上的点的颜色深度随着仿真时间的推移而逐渐减弱。

High quality rendering：高质量绘图复选框。选定后，通过光栅操作绘制高质量的观测仪图，此时星座图观测仪的运行速度较慢；如未选定，通过异或操作快速地绘制质量较低的星座图。

Show grid：网格显示复选框。选定后，在星座图中显示网格线。

3.5.2.3　Axes Properties 类参数项说明

“Axes Properties”用来设定星座图中的坐标轴属性。选定该项后，显示如图 3-22 所示的参数设定框。

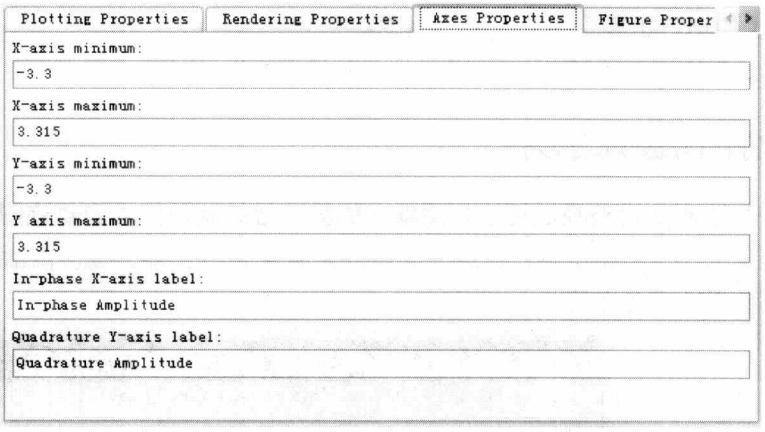


图 3-22　“Axes Properties”参数设定框

由图 3-22 可知，该项包含下面几个参数：

X-axis minimum：设定星座图观测仪横坐标的最小值。

X-axis maximum：设定星座图观测仪横坐标的最大值。

Y-axis minimum：设定星座图观测仪纵坐标的最小值。

Y-axis maximum：设定星座图观测仪纵坐标的最大值。

In-phase X-axis label：设定是否显示横坐标的标签。

Quadrature Y-axis label：设定是否显示纵坐标的标签。

3.5.2.4　Figure Properties 类参数项说明

“Figure Properties”用来设定眼图属性。选定该项后，显示如图 3-23 所示的参数设定框。

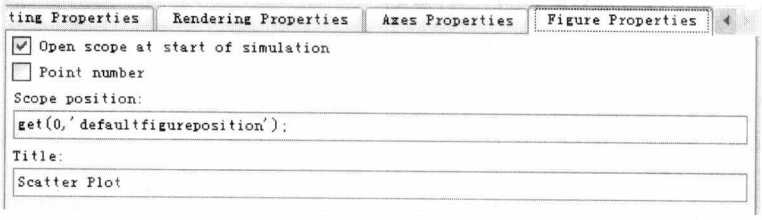


图 3-23 “Figure Properties”参数设定框

由图 3-23 可知，该项包含下面几个参数：

Open scope at start of simulation：本项为复选框。选定后，星座图将在仿真开始的时候自动打开；否则，用户需要双击星座图观测模块之后才能显示。

Point number：设定是否在星座图中显示当前点的编号。

Scope position：设定星座图的位置。它是由[left bottom width height]四个元素组成的向量。left 和 bottom 分别表示星座图左下角的纵坐标和横坐标，如(0,0)表示星座图的左下角。width 和 height 分别表示星座图的宽度和高度。

Title：设定星座图的标题。

3.5.3 离散信号轨迹观测设备

离散信号轨迹观测设备模块可以根据输入的复信号绘制该信号的轨迹图。轨迹图的横坐标是输入复信号的 I 支路分量，纵坐标是输入复信号的 Q 支路分量。离散信号轨迹观测设备模块及其参数设定框如图 3-24 所示。

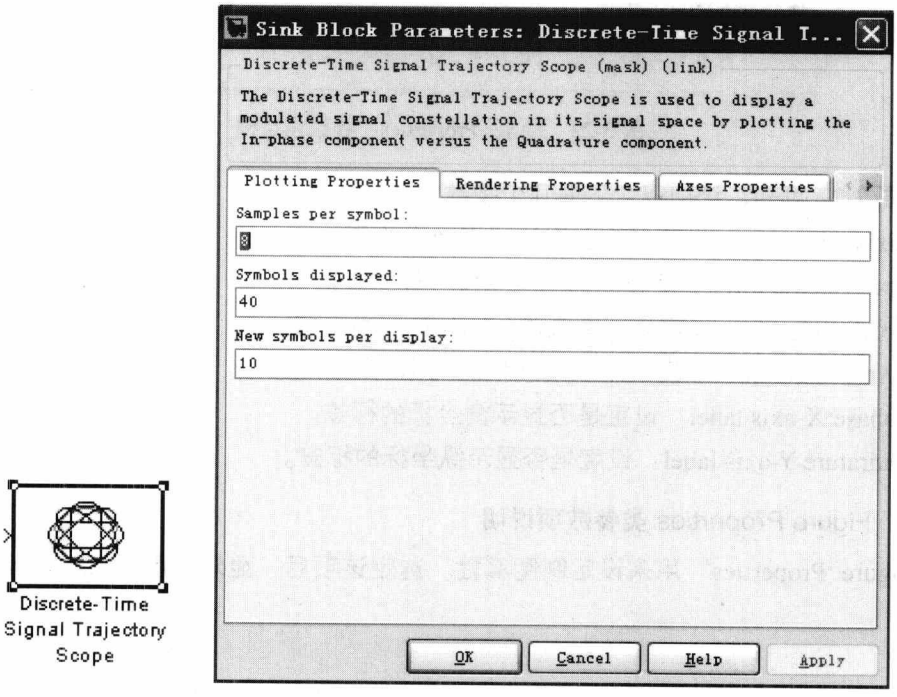


图 3-24 离散信号轨迹观测设备模块及其参数设定框

由图 3-24 可见，离散信号轨迹观测设备参数设定框中又包含四个选项，这是因为离散信号轨迹图需要设定的参数较多，为了方便起见，MATLAB 将其归结为四类，分别为“Plotting Properties”、“Rendering Properties”、“Axes Properties”、“Figure Properties”，默认项为“Plotting Properties”。选定某一类后，参数设定框相应变化。下面分别就这四类做简单的介绍。

3.5.3.1　Plotting Properties 类参数项说明

“Plotting Properties”用来设定离散信号轨迹图的绘制方式。该项为默认项，如图 3-24 所示。其中包含如下参数：

Samples per symbol：设定离散信号轨迹图中每个符号的抽样点数目。

Symbols displayed：设定离散信号轨迹图中显示的符号数目。

New symbols per display：设定离散信号轨迹图每次需要重新设定的点数目。

3.5.3.2　其他类参数项说明

“Rendering Properties”、“Axes Properties”、“Figure Properties”三项的参数设置和星座图观测仪中这三项的设置相同，这里不再赘述。

3.5.4　误码率计算器

误码率计算器模块分别从发射端和间接手段得到输入数据，再对两个数据进行比较，根据比较的结果计算误码率。

应用这个模块，既可以得到错误比特率，也可以得到错误符号率。当输入信号是二进制数据时，则统计的结果是错误比特率，否则，统计得到的结果是错误符号率。误码率计算器模块只比较两个输入信号的正负关系，而不具体地比较它们的大小。误码率计算器模块及其参数设定框如图 3-25 所示。

误码率计算器模块中有若干参数，下面分别对其做简单介绍。

Receive delay：接收端时延设定项。

在通信系统中，接收端需要对接收到的信号进行解调、解码或解交织，这些过程可能会产生一定的时延，使得到达误码率计算器接收端的信号滞后于发送端的信号。为了弥补这种时延，误码率计算器模块需要把发送端的输入数据延迟若干个输入数据，本参数即表示接收端输入的数据滞后发送端数据输入数据的大小。

Computation delay：计算时延设定项。在仿真过程中，有时需要忽略初始的若干输入数据，这就可以通过本项设定。

Computation mode：计算模式项。误码率计算器模块有三种计算模式。分别为帧计算模式、掩码模式、端口模式。其中帧计算模式对发送端和接收端的所有输入数据进行统计。在掩码模式下，模块根据掩码指定对特定的输入数据进行统计，掩码的内容可由参数项“Selected samples from frame”设定。在端口模式下，模块会新增一个输入端口 Sel，只有此端口的输入信号有效时才统计错误率。

Selected samples from frame：掩码设定项。本参数用于设定哪些输入数据需要统计。

本项只有当“Computation mode”项设定为 samples from mask 时有效。

Output data：设定数据输出方式，有 Workspace 和 Port 两种方式。Workspace 时将统计数据输出到工作区。Port 时将统计数据从端口中输出。

Variable name：指定用于保存统计数据的工作区间变量的名称。本项只有在“Output data”设定为 Workspace 时有效。

Reset port：复位端口项。选定此项后，模块增减一个输入端口 Rst，当这个信号有效时，模块被复位，统计值重新设定为 0。

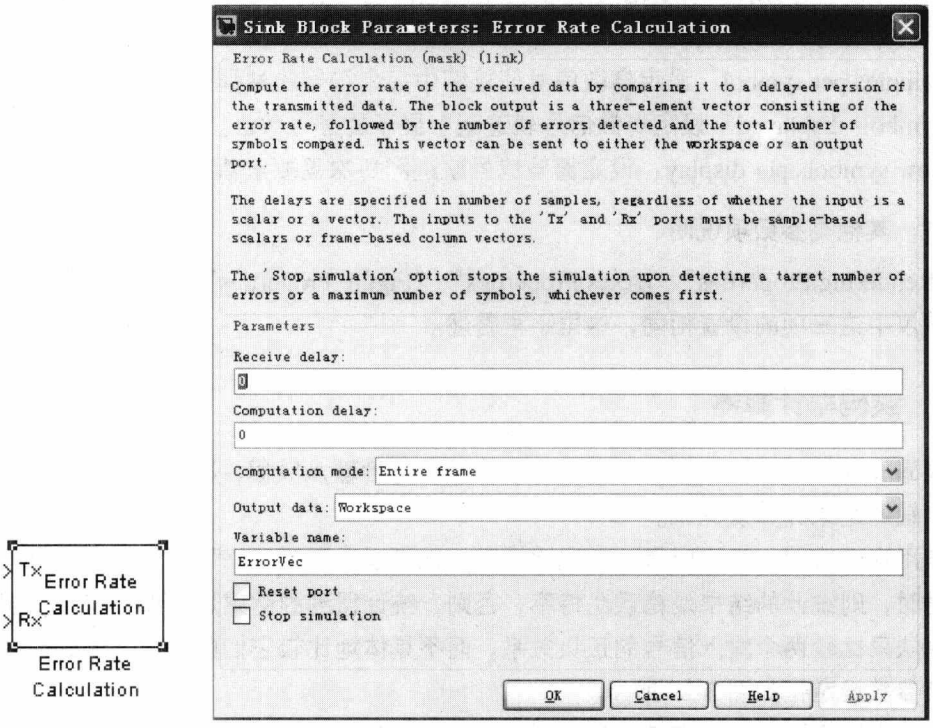


图 3-25　误码率计算器模块及其参数设定框

Stop simulation：仿真停止项。选定本项后，如果模块检测到指定书目的错误，或数据的比较次数达到了门限，则停止仿真过程。

Target number of errors：错误门限项。用于设定仿真停止之前允许出现错误的最大个数。本项只有在“Stop simulation”选定后有效。

Maximum number of symbols：比较门限项。用于设定仿真停止之前允许比较的输入数据的最大个数。本项只有在“Stop simulation”选定后有效。

3.6 本章小结

本章对通信系统中的三大部分：信源、信道和信宿都做了系统的介绍。重点对 Simulink 中三部分的模块分别做了功能阐述及各参数项说明。读者学习时，建议前后比较，以加深学习印象，巩固 Simulink 通信仿真的基础。

第 4 章　信源编码/译码

在现代通信系统中经常使用的数字调制过程中，信源输出的模拟信号，要转换成数字信号，就需要对信源进行编码译码操作。信源编码是用量化的方法将一个源信号转化为一个数字信号，所得信号的符号为某一有限范围内的非负整数。信源译码就是从信源编码的信号恢复到原来的信号。

本章包含两个小节，分别对信源编码和信源译码做简单介绍。

4.1 信源编码

信源编码也称为量化或信号格式化，它一般是为了减少冗余度或为后续的处理做准备而进行的数据处理。在 Simulink 中，包含了 A 律编码、μ 律编码、差分编码和量化编码等若干模块，这里分别进行介绍。

4.1.1 A 律编码

模拟信号的量化有两种方式：均匀量化和非均匀量化。均匀量化把输入信号的取值范围等距离地分割成若干个量化区间，无论抽样值大小如何，量化噪声的均方根固定不变，因此实际过程中大多采用非均匀量化。比较常用的两种非均匀量化的方法是 A 律压缩和 μ 律压缩。美国采用 μ 律压缩和扩展，我国和欧洲各国均采用 A 律压缩和扩展。这里我们先介绍 A 律压缩编码。

如果输入信号为 x，输出信号为 y，则 A 律压缩满足式 4-1。

$$y=\begin{cases}\dfrac{A|x|}{1+\log A}\operatorname{sgn}(x) & 0\leqslant|x|\leqslant\dfrac{V}{A}\\[2ex]\dfrac{V(1+\log(A|x|/V))}{1+\log A}\operatorname{sgn}(x) & \dfrac{V}{A}\leqslant|x|\leqslant V\end{cases}\qquad（式\ 4\text{-}1）$$

式中，A 为 A 律压缩参数，最常采用的 A 值为 87.6；V 为输入信号的峰值；log 为自然对数；sgn 函数当输入为正时，输出 1，当输入为负时，输出 0。

模块的输入并无限制。如果输入为向量，则向量中的每一个分量将会被单独处理。

A 律压缩编码模块及其参数设定框如图 4-1 所示。

A 律压缩编码模块中包含两个参数：

A value：用于指定压缩参数 A 的值。

Peak signal magnitude：用于指定能输入信号的峰值 V。

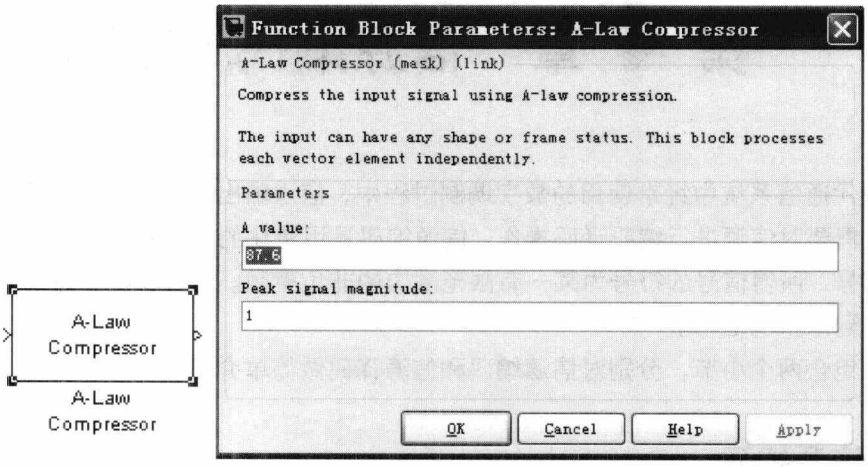


图 4-1　A 律压缩编码模块及其参数设定框

4.1.2　μ 律编码

和 A 律压缩编码类似，μ 律压缩编码中如果输入信号为 x，输出信号为 y，则 μ 律压缩满足式 4-2。

$$y=\frac{V\log(1+\mu|x|/V)}{\log(1+\mu)}\mathrm{sgn}(x) \qquad (式\ 4\text{-}2)$$

式中，μ 为 μ 律压缩参数；V 为输入信号的峰值；log 为自然对数；sgn 函数当输入为正时，输出 1，当输入为负时，输出 0。

模块的输入并无限制。如果输入为向量，则向量中的每一个分量将会被单独处理。

μ 律压缩编码模块及其参数设定框如图 4-2 所示。

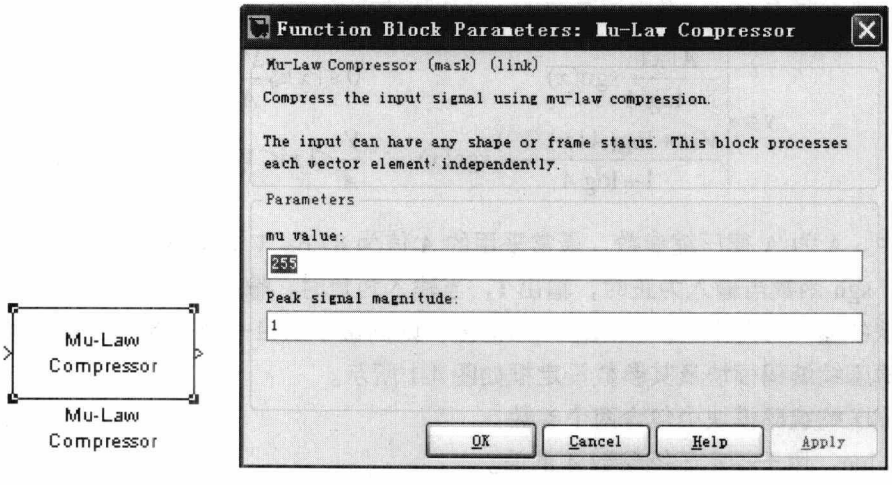


图 4-2　μ 律压缩编码模块及其参数设定框

μ 律压缩编码模块中包含两个参数：

mu value：用于指定 μ 律压缩参数 μ 的值。

Peak signal magnitude：用于指定能输入信号的峰值 V，也是输出信号的峰值。

4.1.3　差分编码

差分编码又称为增量编码，它用一个二进制数来表示前后两个抽样信号之间的大小关系。在 MATLAB 中，差分编码器根据当前时刻之前的所有输入信息计算输出信号，这样，在接收端就可以只按照接收到的前后两个二进制信号恢复出原来的信息序列。

差分编码模块对输入的二进制信号进行差分编码，输出二进制的数据流。输入的信号可以是标量、流向量或帧格式的行向量。如果输入信号为 $m(t)$，输出信号为 $d(t)$，那么 t_k 时刻的输出 $d(t_k)$ 不仅与当前时刻的输入信号 $m(t_k)$ 有关，而且与前一时刻的输出 $d(t_{k-1})$ 有关，如式 4-3 所示。

$$\begin{cases} d(t_0) = (m(t_0) + 1) \bmod 2 \\ d(t_k) = (d(t_{k-1}) + m(t_k) + 1) \bmod 2 \end{cases} \quad (式\ 4\text{-}3)$$

即输出信号 y 取决于当前时刻以及当前时刻之前所有的输入信号的数值。

差分编码模块及其参数设定框如图 4-3 所示。

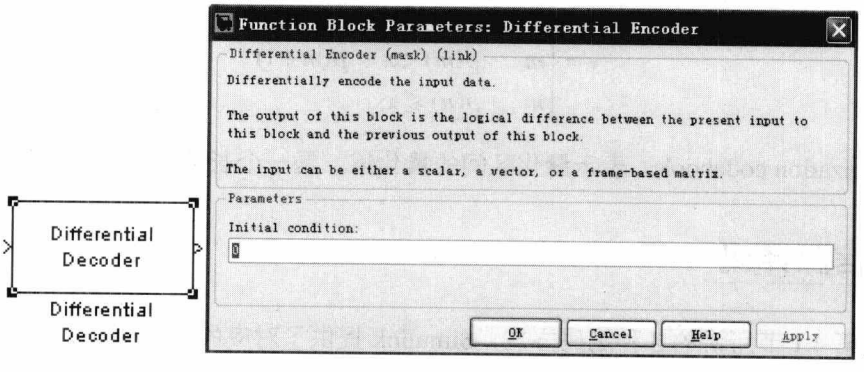


图 4-3　差分编码模块及其参数设定框

差分编码模块中包含一个参数：

Initial condition：用于指定信号符号之间的间隔。

4.1.4　量化编码

量化编码模块用标量量化法来量化输入信号。它根据量化间隔和量化码本把输入信号转换成数字信号，并且输出量化指标、量化电平、编码信号和量化均方误差。

模块的输入信号可以是标量、流向量或矩阵。模块的输入输出信号长度相同。

量化编码模块及其参数设定框如图 4-4 所示。

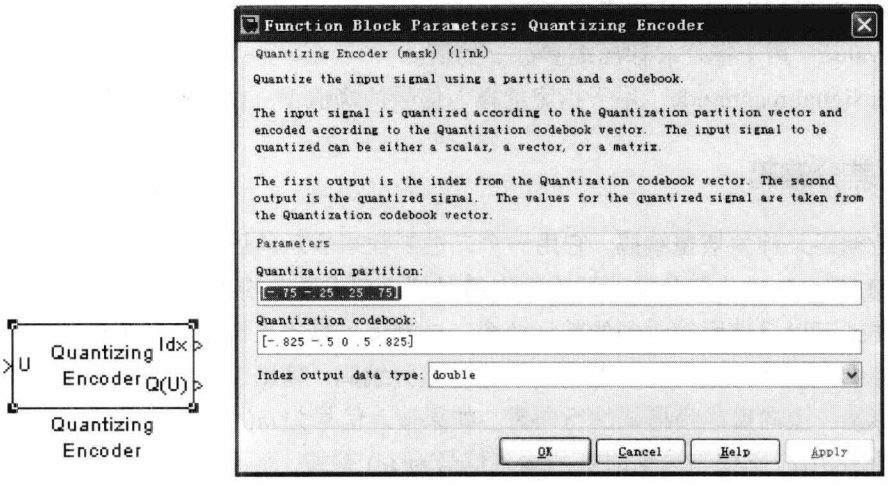


图 4-4　量化编码模块及其参数设定框

量化编码模块中包含两个参数：

Quantization partition：用于指定量化区间，为一个长度为 n 的向量（n 为码元素）。该向量分量要严格按照升序排列。如果设该参量为 p，那么模块的输出 y 与输入 x 之间的关系满足：

$$y=\begin{cases}0 & x\leqslant p(1)\\ m & p(m)<x\leqslant p(m+1)\\ n & p(n)\leqslant x\end{cases}$$

Quantization codebook：表示量化区间的量化值，是一个长度为 n+1 的向量。

4.2 信源译码

参照第 4.1 节所讲的各种编码方式，Simulink 提供了对应的译码模块。

4.2.1 A 律译码

A 律译码模块用来恢复被 A 律压缩编码模块压缩的信号。它的过程与 A 律压缩编码模块正好相反。A 律译码模块的特征函数是 A 律压缩编码模块特征函数的反函数，如式 4-4 所示。

$$x=\begin{cases}\dfrac{y(1+\log A)}{A} & 0\leqslant|y|\leqslant\dfrac{V}{1+\log A}\\ \exp(|y|(1+\log A)/V-1)\dfrac{V}{A}\operatorname{sgn}(y) & \dfrac{V}{1+\log A}\leqslant|y|\leqslant V\end{cases}\qquad（式\ 4\text{-}4）$$

A 律译码模块及其参数设定框如图 4-5 所示。

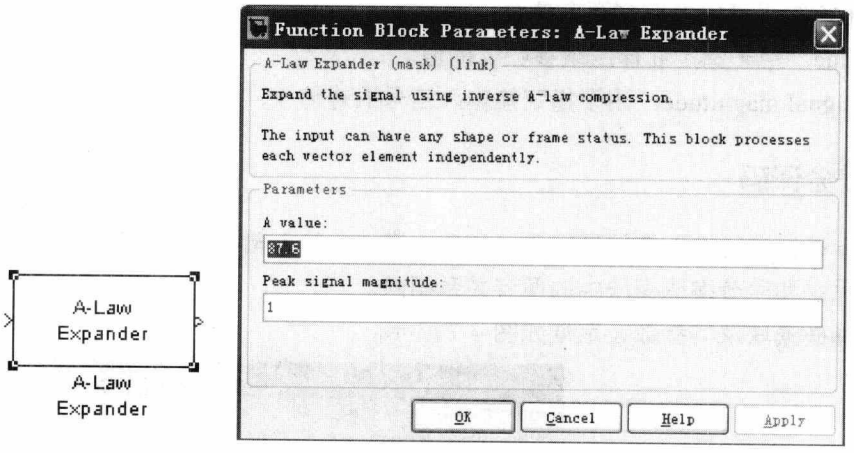


图 4-5　A 律译码模块及其参数设定框

A 律译码模块中包含两个参数：

A value：用于指定压缩参数 A 的值。

Peak signal magnitude：用于指定能输入信号的峰值 V。同时也是输出信号的峰值。

4.2.2　μ 律译码

μ 律译码模块用来恢复被 μ 律压缩编码模块压缩的信号。它的过程与 μ 律压缩编码模块正好相反。μ 律译码模块的特征函数是 μ 律压缩编码模块特征函数的反函数，如式 4-5 所示。

$$x=\frac{V}{\mu}(e^{|y|\log(1+\mu)/V}-1)\operatorname{sgn}(y) \qquad (式\ 4\text{-}5)$$

μ 律译码模块及其参数设定框如图 4-6 所示。

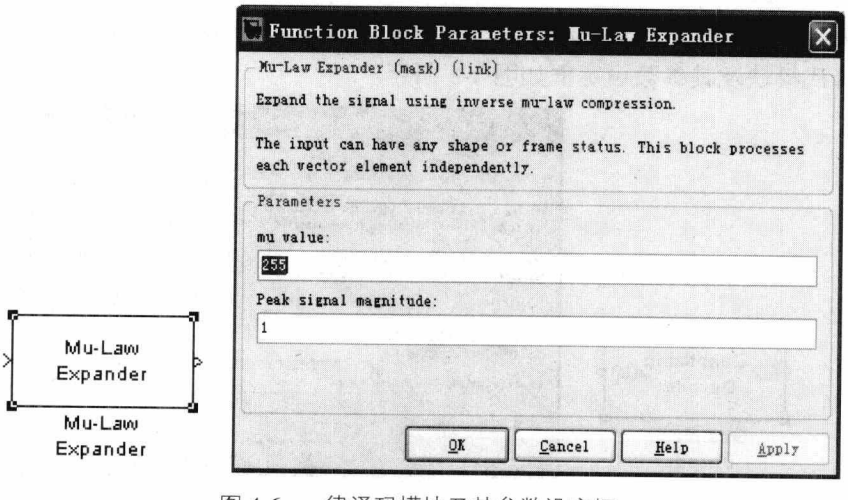


图 4-6　μ 律译码模块及其参数设定框

μ 律压缩编码模块中包含两个参数：

mu value：用于指定 μ 律压缩参数 μ 的值。

Peak signal magnitude：用于指定能输入信号的峰值 V，也是输出信号的峰值。

4.2.3 差分译码

差分译码模块对输入信号进行差分译码。模块的输入输出均为二进制信号，且输入输出之间的关系和差分编码模块中的两者关系相同。

差分译码模块及其参数设定框如图 4-7 所示。

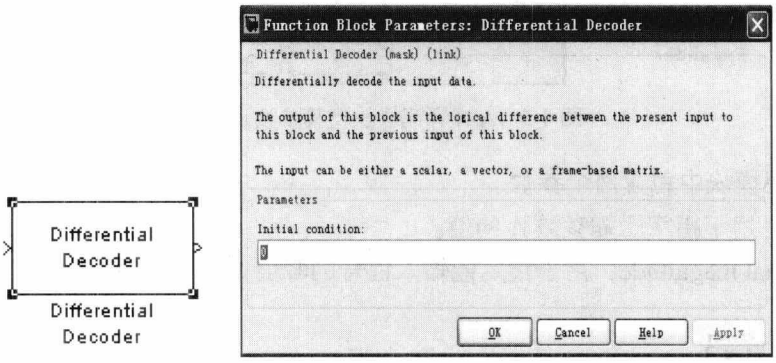


图 4-7 差分译码模块及其参数设定框

差分译码模块中包含一个参数：

Initial condition：用于指定信号符号之间的间隔。

4.2.4 量化译码

量化译码模块用于从量化信号中恢复出消息，它执行的是量化编码模块的逆过程。模块的输入信号是量化的区间号，可以是标量、流向量或矩阵。如果输入为向量，那么向量的每一个分量将被分别单独处理。量化译码模块中的输入输出信号的长度相同。

量化译码模块及其参数设定框如图 4-8 所示。

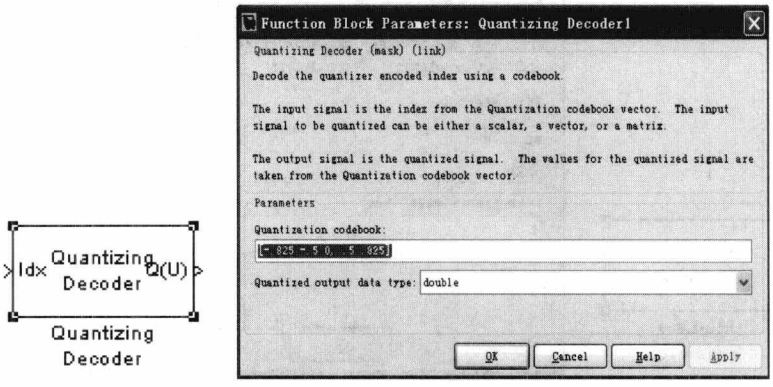


图 4-8 量化译码模块及其参数设定框

量化译码模块中包含一个参数：

Quantization codebook：表示每一个非负整数输入所对应的输出实向量。

4.3 本章小结

本章介绍了信源编码译码的知识，读者通过学习，可以了解 MATLAB 中提供的四种编码/译码模块：A 律编码/译码模块、μ 律编码/译码模块、差分编码/译码模块和量化编码/译码模块的功能、参数项，熟悉信源编码/译码的原理和特点。

第 5 章　调制与解调

在现有大多数的通信介质中，往往只能传输某一特定的频率范围。如果要传输一个频率在信道通带范围之外的信号，就需要利用某些手段根据输入信号改变传输信号，这种改变就称为调制。解调是调制的逆过程，是把某种特定形态的传输波形还原为发送端调制前的信号。

调制与解调是通信系统中十分重要的环节，针对不同的信道环境选择不同的调制与解调方式可以有效地提高通信系统中的频带利用率。调制解调技术按照通信信号的类型可以分为模拟调制解调和数字调制解调。而两种形式都可以通过幅度调制、频率调制和相位调制实现。对调制解调过程的仿真也有两种：通带仿真和基带仿真。由于在较高信号频率下，通带仿真的速度较慢，而且效率不高，一般用基带仿真代替通带仿真。

依据上面的分类，本章分为两大部分，分别介绍模拟调制解调和数字基带调制解调，每一部分又通过幅度调制、频率调制和相位调制等小节做详细说明。

5.1 模拟调制解调

MATLAB 中提供了多个模拟调制解调的模块，下面分别做详细介绍。

5.1.1 DSB AM 调制解调

5.1.1.1 DSB AM 调制

DSB AM 调制模块对输入信号进行双边带幅度调制。输出为通带表示的调制信号。输入和输出信号都是基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$(u(t)+k)\cos(2\pi f_c t+\theta)$。其中 k 为“Input signal offset”参数，f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。通常设定 k 为输入信号 $u(t)$负值部分最小值的绝对值。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

DSB AM 调制模块及其参数设定项如图 5-1 所示。

DSB AM 调制模块中包含下面几个参数项：

Input signal offset：设定补偿因子 k，应该大于等于输入信号最小值的绝对值。

Carrier frequency (Hz)：设定载波频率。

Initial phase (rad)：设定载波初始相位。

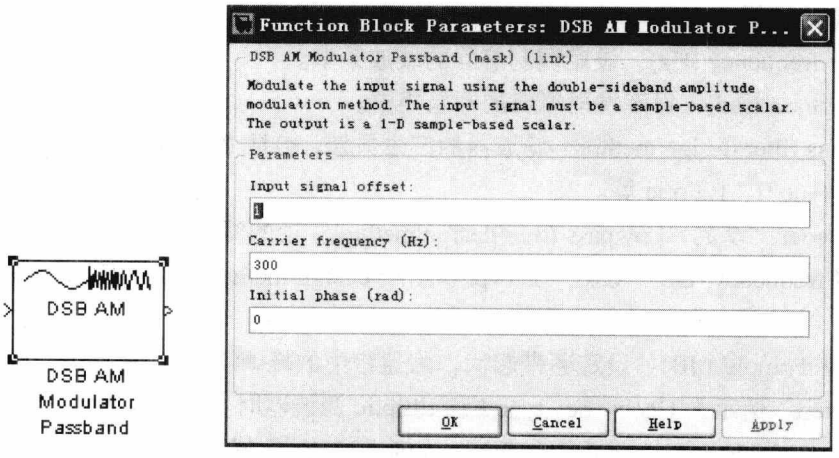


图 5-1　DSB AM 调制模块及其参数设定项

5.1.1.2　DSB AM 解调

DSB AM 解调模块对双边带幅度调制的信号进行解调。输入信号为通带表示的调制信号，且输入输出信号均为基于采样的实数标量信号。

在解调过程中，DSB AM 解调模块使用了低通滤波器。在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

DSB AM 解调模块及其参数设定框如图 5-2 所示。

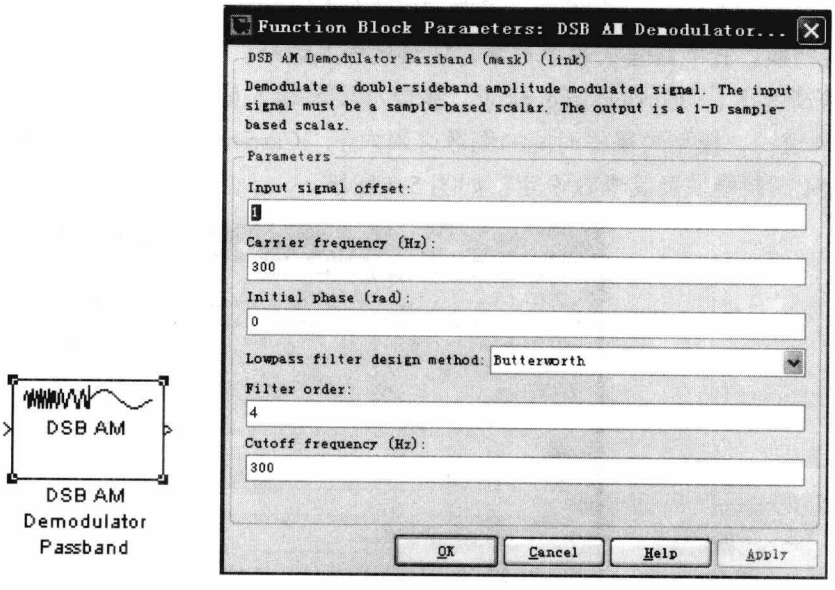


图 5-2　DSB AM 解调模块及其参数设定框

DSB AM 解调模块包含下面几个参数项：

Input signal offset：设定输出信号偏移。模块中的所有解调信号都将减去这个偏移量，

从而得到输出数据。

Carrier frequency (Hz)：设定调制信号的载波频率。

Initial phase (rad)：设定发射载波的初始相位。

Lowpass filter design method：滤波器的产生方法，包括 Butterworth、Chebyshev type I、Chebyshev type II、Elliptic 等。

Filter order：设定“Lowpass filter design method”项的滤波阶数。

Cutoff frequency (Hz)：设定“Lowpass filter design method”项的低通滤波器的截止频率。

Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type I 和 Elliptic 滤波器时，本项有效。

Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type II 和 Elliptic 滤波器时，本项有效。

5.1.2 SSB AM 调制解调

5.1.2.1 SSB AM 调制

SSB AM 调制模块使用希尔伯特滤波器进行单边带幅度调制。输出为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$u(t)\cos(f_c t+\theta)\mp\hat{u}(t)\sin(f_c t+\theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。$\hat{u}(t)$ 表示输入信号的 $u(t)$ 的希尔伯特转换。式中减号代表上边带，加号代表下边带。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

SSB AM 调制模块及其参数设定项如图 5-3 所示。

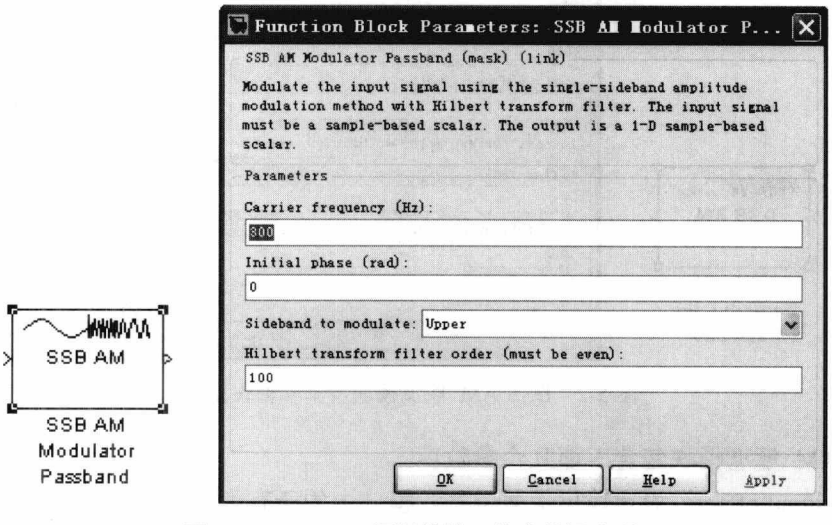


图 5-3 SSB AM 调制模块及其参数设定项

SSB AM 调制模块包含下面几个参数项：

Carrier frequency (Hz)：设定载波频率。

Initial phase (rad)：已调制信号的相位补偿 θ。

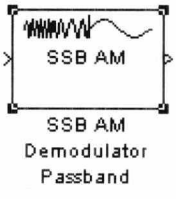
Sideband to modulate：传输方式设定项。有 upper 和 lower 两种，分别为上边带传输和下边带传输。

Hilbert Transform filter order：设定用于希尔伯特转化的 FIR 滤波器的长度。

5.1.2.2　SSB AM 解调

SSB AM 解调模块对单边带幅度调制信号进行解调。输入为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

SSB AM 解调模块及其参数设定框如图 5-4 所示。



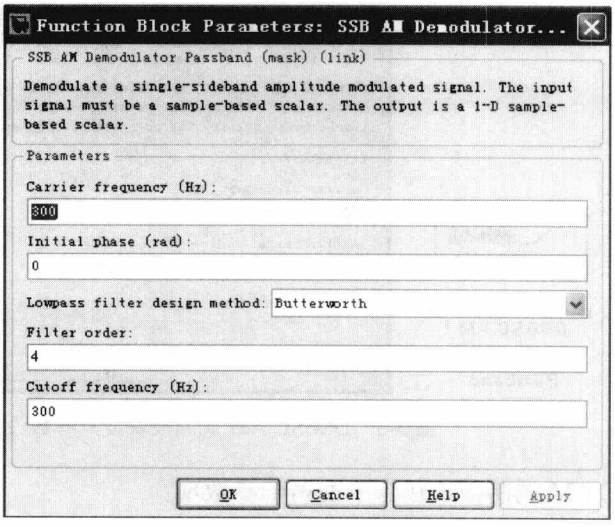


图 5-4　SSB AM 解调模块及参数设定框

SSB AM 解调模块包含下面几个参数项：

Carrier frequency (Hz)：SSB AM 解调模块中调制信号的载波频率。

Initial phase (rad)：已调制信号的相位补偿 θ。

Lowpass filter design method：滤波器的产生方法，包括 Butterworth、Chebyshev type I、Chebyshev type II、Elliptic 等。

Filter order：设定“Lowpass filter design method”项中选定的数字低通滤波器的滤波阶数。

Cutoff frequency (Hz)：设定“Lowpass filter design method”项的数字低通滤波器的截止频率。

Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type I 和 Elliptic 滤波器时，本项有效。

Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type II 和 Elliptic 滤波器时，本项有效。

5.1.3 DSBSC AM 调制解调

5.1.3.1 DSBSC AM 调制

DSBSC AM 调制模块进行双边带一致载波幅度调制。输出信号为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$u(t)\cos(f_c t+\theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

DSBSC AM 调制模块及其参数设定框如图 5-5 所示。

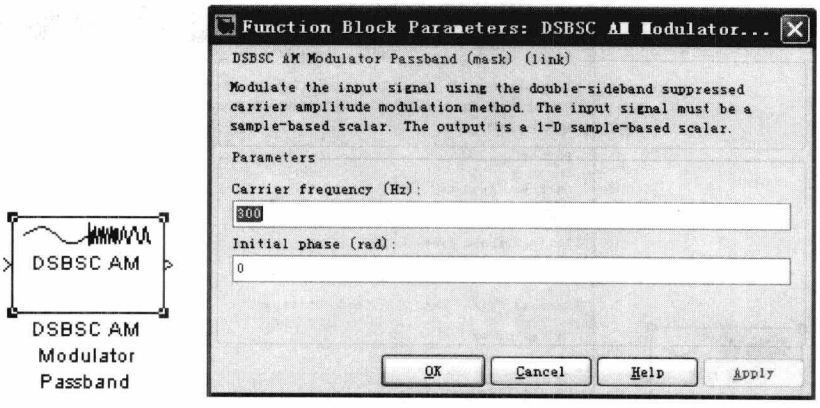


图 5-5 DSBSC AM 调制模块及其参数设定框

DSBSC AM 调制模块包含下面两个参数项：

Carrier frequency (Hz)：设定载波频率。

Initial phase (rad)：设定初始相位的载波频率。

5.1.3.2 DSBSC AM 解调

DSBSC AM 解调模块对双边带抑制载波幅度调制信号进行解调。输入信号为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

DSBSC AM 解调模块及其参数设定框如图 5-6 所示。

DSBSC AM 解调模块中包含下面几个参数项：

Carrier frequency (Hz)：DSBSC AM 解调模块中调制信号的载波频率。

Initial phase (rad)：设定载波初始相位。

Lowpass filter design method；滤波器的产生方法，包括 Butterworth、Chebyshev type I、Chebyshev type II、Elliptic 等。

Filter order：设定“Lowpass filter design method”项中选定的数字低通滤波器的滤波

阶数。

Cutoff frequency (Hz)：设定“Lowpass filter design method”项的数字低通滤波器的截止频率。

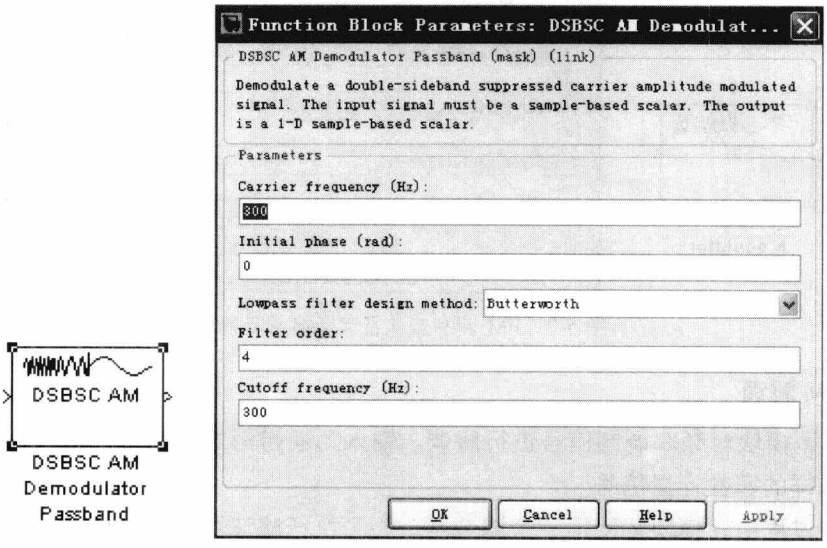


图 5-6　DSBSC AM 解调模块及其参数设定框

Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type I 和 Elliptic 滤波器时，本项有效。

Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev type II 和 Elliptic 滤波器时，本项有效。

5.1.4　FM 调制解调

5.1.4.1　FM 调制

FM 调制模块用于频率调制。输出为通带形式的调制信号。输出信号的频率随着输入信号的幅度而变化，输入和输出信号均采用基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$\cos(2\pi f_c t + 2\pi K_c \int_0^t u(\tau)d\tau + \theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。K_c 为“Modulation constant”参数。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

FM 调制模块及其参数设定框如图 5-7 所示。

FM 调制模块包括下面几个参数项：

Carrier frequency (Hz)：表示调制信号的载波频率。

Initial phase (rad)：表示发射载波的初始相位。

Frequency deviation (Hz)：表示载波频率的频率偏移。

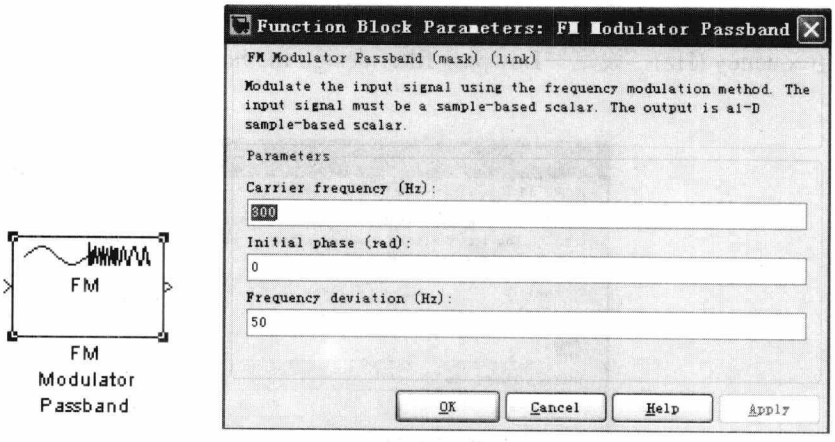


图 5-7　FM 调制模块及其参数设定框

5.1.4.2　FM 解调

FM 解调模块对频率调制信号进行解调。输入为通带形式的信号，输入和输出信号均采用基于采样的实数标量信号。

在解调过程中，模块要使用一个滤波器。为了执行滤波器的希尔伯特转化，载波频率最好大于输入信号采样时间的 10%。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

FM 解调模块及其参数设定框如图 5-8 所示。

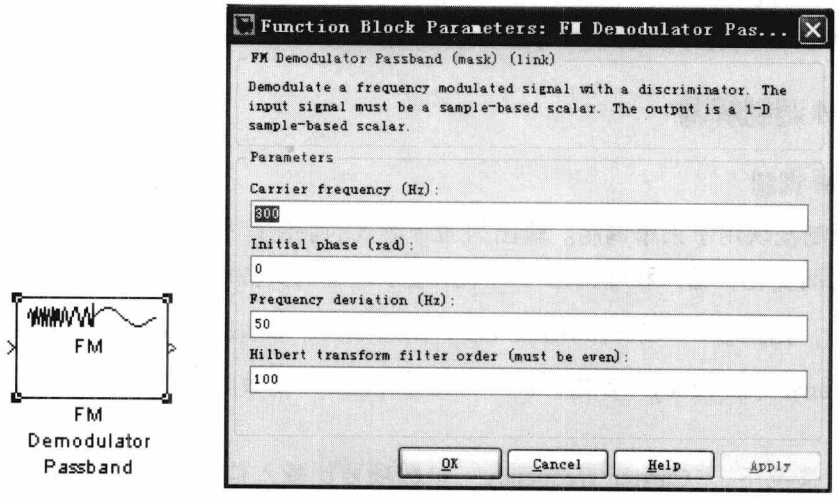


图 5-8　FM 解调模块及其参数设定框

FM 解调模块中包含下面几个参数项：

Carrier frequency (Hz)：表示调制信号的载波频率。

Initial phase (rad)：表示发射载波的初始相位。

Frequency deviation (Hz)：表示载波频率的频率偏移。

Hilbert transform filter order：表示用于希尔伯特转化的 FIR 滤波器的长度。

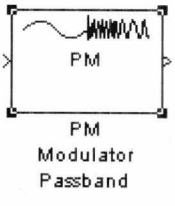
5.1.5　PM 调制解调

5.1.5.1　PM 调制

PM 调制模块进行通带相位调制。输出为通带表示的调制信号。输出信号的频率随输入幅度变化而变化。输入和输出信号均采用基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$\cos(2\pi f_c t + K_c u(t) + \theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。K_c 为“Modulation constant”参数。

PM 调制模块及其参数设定框如图 5-9 所示。



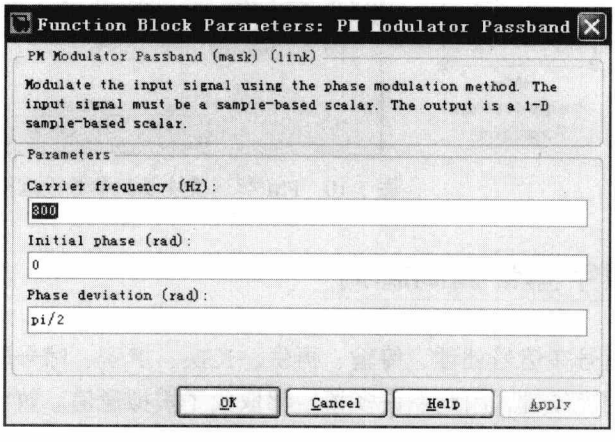


图 5-9　PM 调制模块及其参数设定框

PM 调制模块包含下面几个参数项：

Carrier frequency (Hz)：表示调制信号的载波频率。

Initial phase (rad)：表示发射载波的初始相位。

Frequency deviation (Hz)：表示载波频率的频率偏移。

5.1.5.2　PM 解调

PM 解调模块对通带相位调制的信号进行解调。输入信号为通带形式的已调信号，输入和输出均为基于采样的实数标量信号。

在解调过程中，模块要使用一个滤波器。为了执行滤波器的希尔伯特转化，载波频率最好大于输入信号采样时间的 10%。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的两倍。

PM 解调模块及其参数设定框如图 5-10 所示。

PM 解调模块包含下面几个参数项：

Carrier frequency (Hz)：表示调制信号的载波频率。

Initial phase (rad)：表示发射载波的初始相位。

Phase deviation (Hz)：表示载波信号的相位偏移。

Hilbert transform filter order：表示用于希尔伯特转化的 FIR 滤波器的长度。

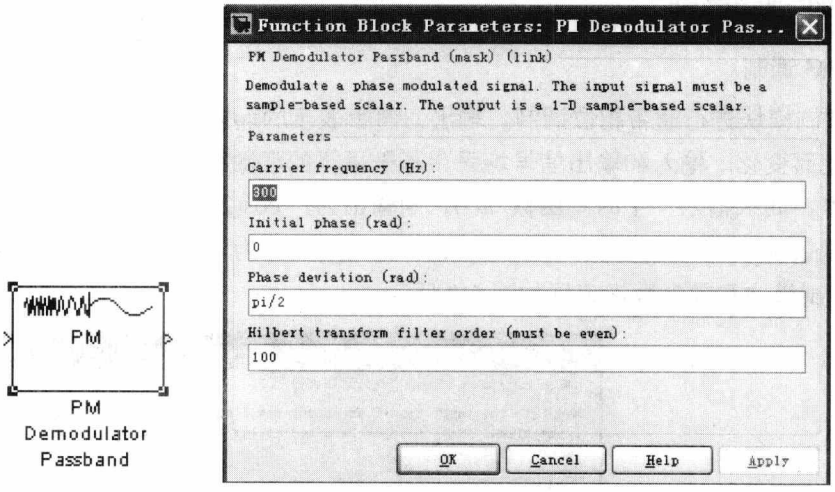


图 5-10　PM 解调模块及其参数设定框

5.2 数字基带调制解调

数字信号在信号处理、传输、再生、交换、加密、信号质量等众多方面有着模拟信号无法比拟的优越性，因此在许多领域都取代了模拟通信。数字调制又可以分成基带调制和频带调制。把频谱从零开始而未经调制的数字信号所占有的频率范围叫做基带频率，简称基带。利用基带信号直接传输的方式称为基带传输。

在 MATLAB 中，提供了多个数字基带调制与解调模块，和模拟调制类似，数字基带调制与解调也可分成幅度调制、频率调制和相位调制三种情况。本节将分别对三种情况下的调制解调模块做简单介绍。

5.2.1 数字幅度调制解调

5.2.1.1 数字幅度调制

MATLAB 对数字幅度调制提供了 General QAM Modulator Baseband、M-PAM Modulator Baseband、Rectangular QAM Modulator Baseband 等多个模块。下面以 M-PAM Modulator Baseband 为例进行介绍。

M-PAM Modulator Baseband 称为 M 相基带幅度调制模块，该模块用于基带 M 元脉冲的幅度调制。模块的输出为基带形式的已调制的信号。模块中“M-ary number”项的参数 M 为信号星座图的点数，而且必须是偶数。

模块使用默认的星座图映射方式，将位于 0～(M-1)的整数 X 映射为复数值[$2X$-M+1]。模块的输入和输出都是离散信号，参数项“Input type”将会决定模块是接收 0～(M-1)的整数，还是接收二进制形式表示的整数。

如果“Input type”设置为 Integer，那么模块接收整数，输入可以是标量，也可以是 int8、uint8、int16、uint16、int32、uint32、single 或者 double 类型的基于帧的列向量。

如果“Input type”设置为 Bit，那么模块接收 K 比特的组，称为二进制字。输入可以是长度为 K 的向量，也可以是长度为 K 的整数倍的基于帧的列向量。在这种情况下，模块可以接受 int8、uint8、int16、uint16、int32、uint32、boolean、single 或 double 类型的数据。

参数“Constellation ordering”决定模块如何将二进制字分配到信号星座图的点。如果此项设为 Binary，那么模块使用自然二进制编码星座图。如果此项设置为 Gray，那么模块使用格雷码星座图。

M-PAM 调制模块及其参数设定项如图 5-11 所示。

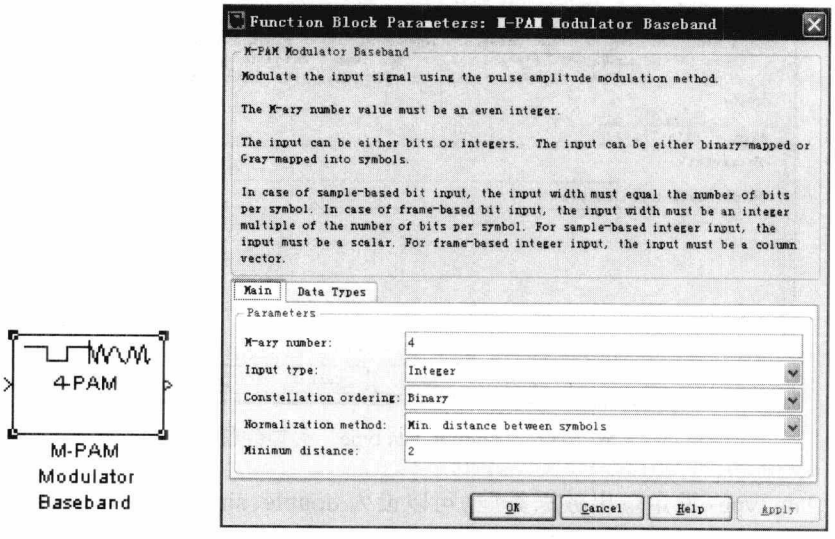


图 5-11　M-PAM 调制模块及其参数设定项

如图 5-11 所示，M-PAM 调制模块参数设定框中包含“Main”和“Data Types”两类，默认为“Main”类。下面分别对各类中的参数项做简单介绍。

1.“Main”类参数项说明

M-ary number：表示信号星座图的点数，该项必须设为一个偶数。

Input type：表示输入是由整数还是由比特组组成。如果该项设为 Bit，那么“M-ary number”项必须为 2^K，其中 K 为正整数。

Constellation ordering：该项决定如何将输入的比特组映射成相应的整数。

Normalization method：为一复选框，决定如何测量信号的星座图，有 Min.distance between symbols、Average Power 和 Peak Power 等可选项。

Minimum distance：表示星座图中两个距离最近点之间的距离。本项只有当“Normalization method”选为 Min.distance between symbols 时有效。

Average power (watts)：星座图中符号的平均功率，本项只有当“Normalization method”选为 Average Power 时有效。

Peak power (watts)：星座图中符号的最大功率。本项只有当“Normalization method”选为 Peak Power 时有效。

2.“Output data type”类参数项说明

“Output data type”类参数项如图 5-12 所示。

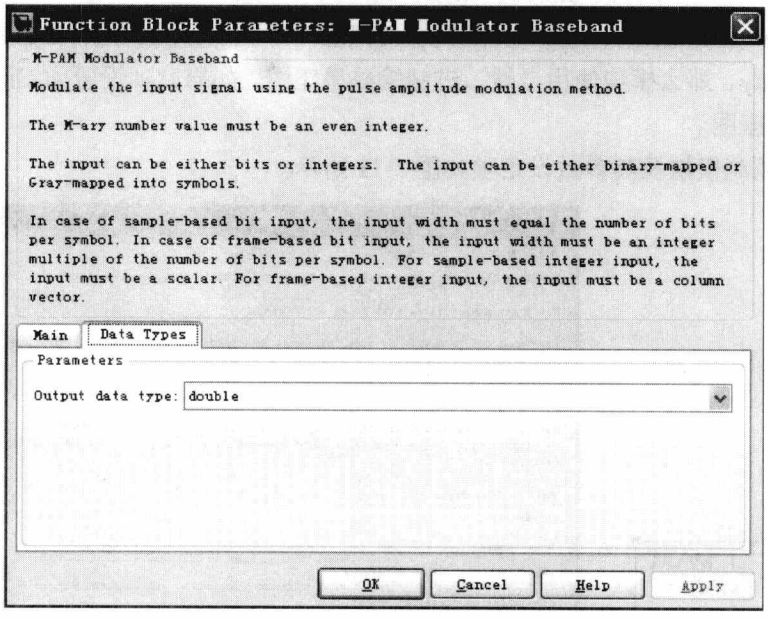


图 5-12 “Output data type”类参数项

Output data type：设定输出数据类型。可以设为 double、single、Fixed-point、User-defined 或 Inherit via back propagation 等多种类型。

Output word length：设定 fixed-point 输出类型的输出字长。本项只有当“Output data type”设为 Fixed-point 时有效并可见。

User-defined data type：设定带符号的内置或定点数据类型。本项只有当“Output data type”设为 User-defined 时有效并可见。

Set output fraction length to：设定固定点输出比例。本项只有当“Output data type”设为 Fixed-point 或 User-defined 项为固定点数据类型时有效并可见。

Output fraction length：设定固定点输出数据的分数位数。

5.2.1.2 数字幅度解调

MATLAB 中对数字幅度解调提供了 General QAM Demodulator Baseband、M-PAM Demodulator Baseband、Rectangular QAM Demodulator Baseband 等多个模块。下面以 M-PAM Demodulator Baseband 为例进行介绍。

M-PAM Demodulator Baseband 称为 M 相基带幅度解调模块，该模块用于基带 M 元脉冲幅度调制的解调。模块的输入为基带形式的已调制信号。

参数项“Output type”将会决定模块是产生整数，还是二进制形式表示的整数。如果

“Output type”设置为 Integer，那么模块输出整数。如果“Output type”设置为 Bit，那么模块输出 K 比特的组，称为二进制字。参数“Constellation ordering”决定模块如何将二进制字分配到信号星座图的点。

M-PAM 解调模块及其参数设定框如图 5-13 所示。

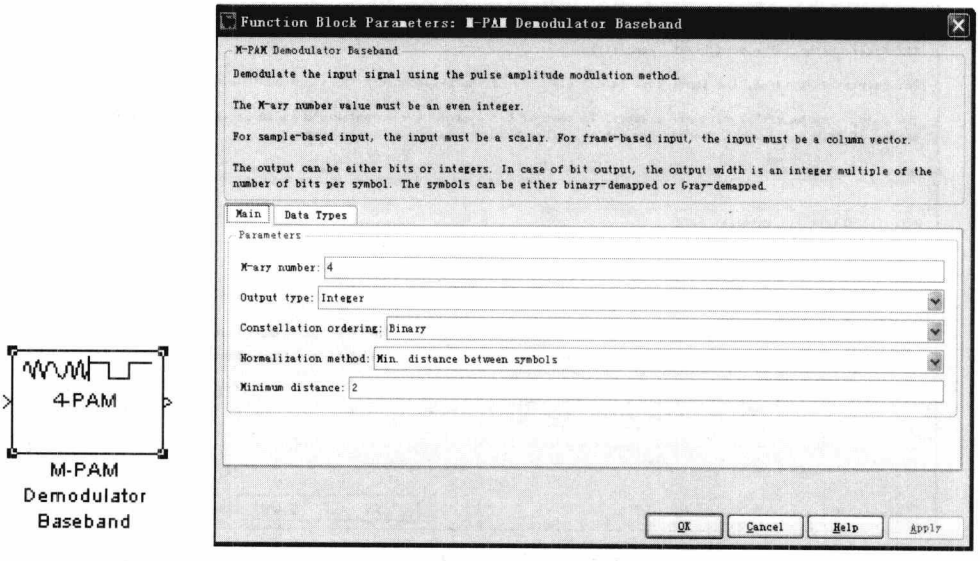


图 5-13　M-PAM 解调模块及其参数设定框

如图 5-13 所示，M-PAM 解调模块参数设定框中包含“Main”和“Data Types”两类，默认为“Main”类，如图 5-11 所示。下面分别对各类中的参数项做简单介绍。

1.“Main”类参数项说明

M-ary number：表示信号星座图的点数，该项必须设为一个偶数。

Output type：表示输出是由整数还是由比特组组成。如果该项设为 Bit，那么“M-ary number”项必须为 2^K，其中 K 为正整数。

Constellation ordering：该项决定如何将输出的比特组映射成相应的整数。本项只有在“Output type”设定为 Bit 时有效。

Normalization method：为一复选框，决定如何测量信号的星座图，有 Min.distance between symbols、Average Power 和 Peak Power 等可选项。

Minimum distance：表示星座图中两个距离最近点之间的距离。本项只有当“Normalization method”选为 Min.distance between symbols 时有效。

Average power (watts)：星座图中符号的平均功率，本项只有当“Normalization method”选为 Average Power 时有效。

Peak power (watts)：星座图中符号的最大功率。本项只有当“Normalization method”选为 Peak Power 时有效。

2．“Output data type”类参数项说明

M-PAM 解调模块的“Output data type”类参数项如图 5-14 所示。

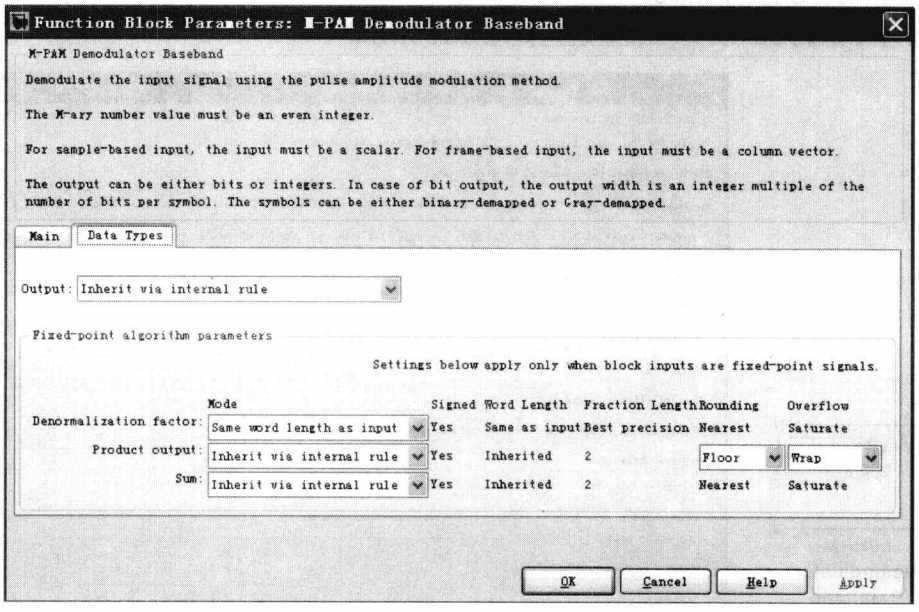


图 5-14　模块的“Output data type”类参数项

Output：输出设定项。当参数设定为 Inherit via internal rule（默认）时，模块的输出数据类型由输入端决定。当输入数据为 single 或 double 类型时，输出与输入类型相同。否则输出数据类型将会和本项设定为 Smallest unsigned integer 情况相同。

当参数设定为 Smallest unsigned integer 时，输出数据的类型由模型中结构参数对话框中的“Hardware Implementation”项决定。如果“Hardware Implementation”项选为 ASIC/FPGA，输出数据类型为 ideal minimum size。如果“Hardware Implementation”项选为其他情况时，输出为满足期望最小长度的最小字长无符号整数。

Denormalization factor：为一复选框，可以选定为 Same word length as input 或者 Specify word length，选定后将会出现一个输入框。

Product output：为一复选框，可以选定为 Inherit via internal rule 或者 Specify word length，选定后将会出现一个输入框。具体可参见 MATLAB help。

Sum：为一复选框，可以选定为 Inherit via internal rule、Same as product output 或者 Specify word length，选定后将会出现一个输入框。具体可参见 MATLAB help。

5.2.2　数字频率调制解调

5.2.2.1　数字频率调制

MATLAB 中提供了 M-FSK Modulator Baseband 模块。该模块进行基带 M 元频移键控调制。输出为基带形式的已调信号。

“M-ary number”项参数 M 为已调信号频率。参数“Frequency separation”为已调信号连续频率之间的间隔。

模块的输入和输出为离散信号。“Input type”项决定模块是接收 0 到 M-1 之间的整数，还是二进制形式的整数。

如果“Input type”项选为 Integer，那么模块接收整数输入。输入可以是标量，也可以是基于帧的列向量。如果“Input type”项选为 Bit，那么模块接收 K 比特的组，称为二进制字。输入可以是长度为 K 的向量或基于帧的列向量（长度为 K 的整数倍）。

M-FSK 调制模块及其参数设定框如图 5-15 所示。

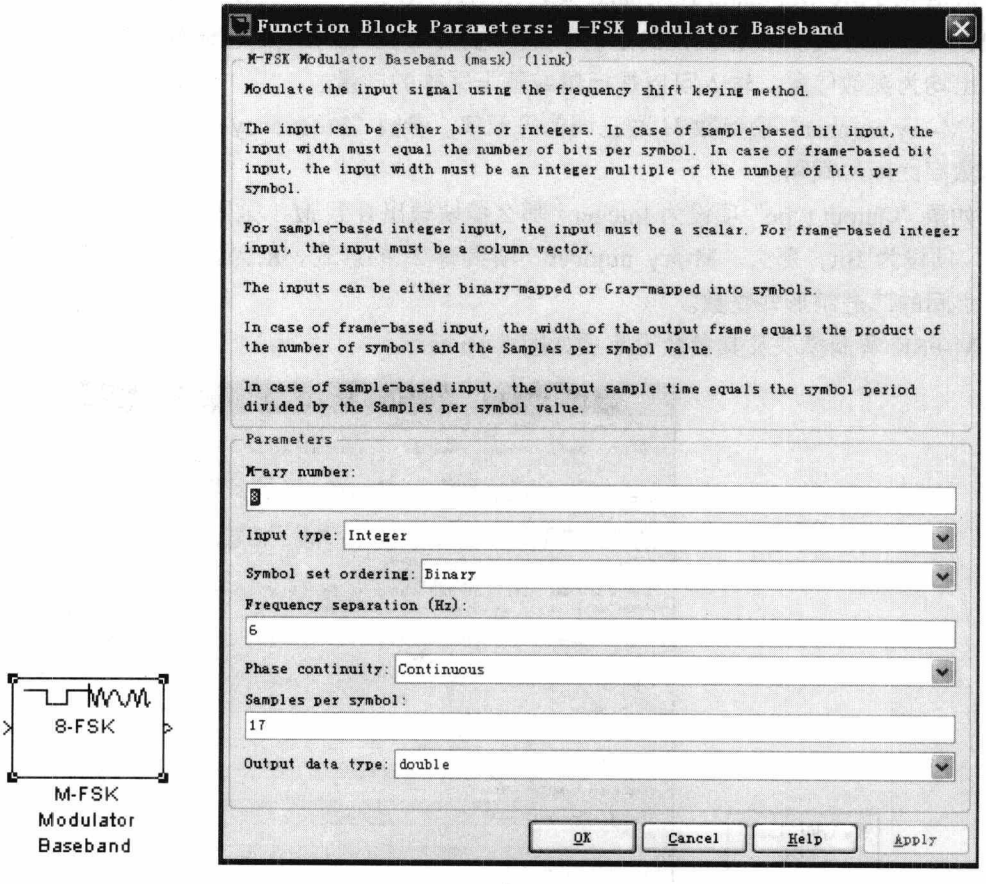


图 5-15　M-FSK 调制模块及其参数设定框

M-FSK 调制模块包含下面几个参数项：

M-ary number：表示信号星座图的点数，M 必须为一个偶数。

Input type：表示输入由整数组成还是由比特组组成。如果本项设为 Bit，那么参数“M-ary number”必须为 2^K，K 为正整数。

Symbol set ordering：设定模块如何将每一个输入比特组映射到相应的整数。

Frequency separation (Hz)：表示已调信号中相邻频率之间的间隔。

Phase continuity：决定已调制信号的相位是连续的还是非连续的。如果本项设为

Continuous，那么即使频率发生变化，调制信号的相位依然维持不变。如果该项设为Discontinuous，那么调制信号由不同频率的 M 正弦曲线部分构成，这样的话如果输入值发生变化，那么调制信号的相位也会发生变化。

Samples per symbol：对应于每个输入的整数或二进制字模块输出的采样个数。

Output data type：设定模块的输出数据类型，可以为 double 或 single。默认为 double 类型。

5.2.2.2 数字频率解调

对应 M-FSK Modulator Baseband 模块，MATLAB 提供了 M-FSK Demodulator Baseband 模块，用于基带 M 元频移键控的解调。模块的输入为基带形式的已调制信号。模块的输入和输出均为离散信号。输入可以是标量或基于采样的向量。

“M-ary number”项参数 M 为已调信号频率。参数“Frequency separation”为已调信号连续频率之间的间隔。

如果“Output type”项设为 Integer，那么模块输出 0 到 M-1 范围的整数。如果“Output type”项设为 Bit，那么“M-ary number”项具有 2^K 的形式，K 为正整数。模块输出 0 到 M-1 之间的二进制形式整数。

M-FSK 解调模块及其参数设定项如图 5-16 所示。

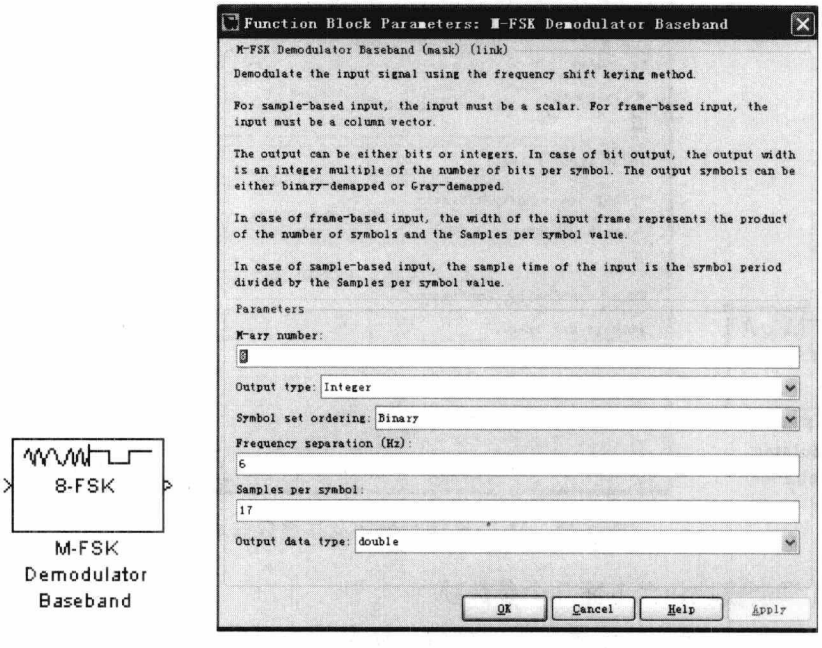


图 5-16 M-FSK 解调模块及其参数设定项

M-FSK 解调模块包含下面几个参数项：

M-ary number：表示信号星座图的点数，M 为一个偶数。

Output type：表示输出数据由整数组成还是由比特组组成。如果本项设为 Bit，那么参数“M-ary number”必须为 2^K，K 为正整数。

Symbol set ordering：设定模块如何将每一个输出比特组映射到相应的整数。

Frequency separation (Hz)：表示已调信号中相邻频率之间的间隔。

Samples per symbol：对应于每个输出的整数或二进制字模块输出的采样个数。

Output data type：设定模块的输出数据类型，可以为 boolean、int8、uint8、int16、uint16、int32、uint32 或 double。默认为 double 类型。

5.2.3　数字相位调制解调

5.2.3.1　数字相位调制

MATLAB 中提供了众多的相位调制解调模块，本节我们以 M-PSK Modulator Baseband 模块为例，简单介绍基带数字相位调制。

M-PSK 调制模块进行基带 M 元相移键控调制。输出为基带形式的已调信号。“M-ary number”项参数 M 表示信号星座图的点数。

M-PSK 调制模块及其参数设定项如图 5-17 所示。

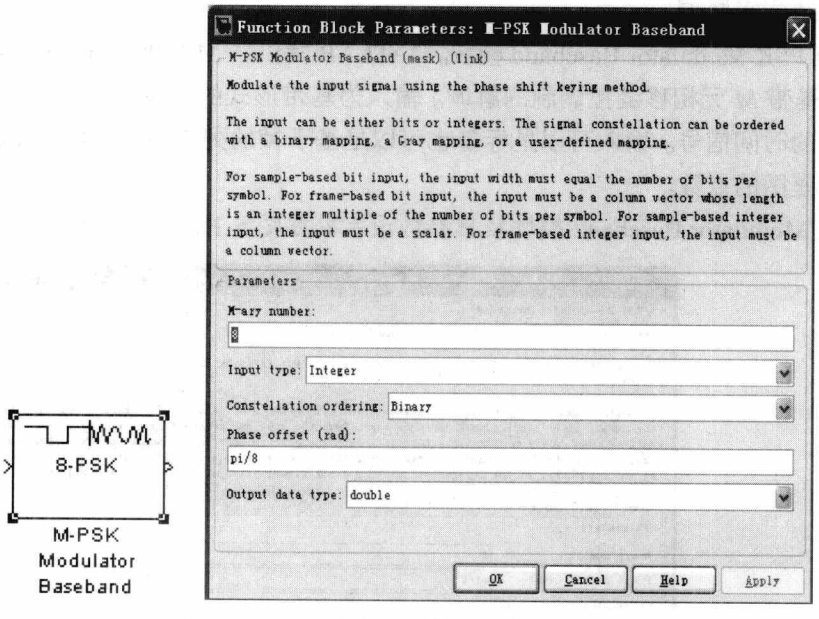


图 5-17　M-PSK 调制模块及其参数设定项

M-PSK 调制模块中包含下面几个参数项：

M-ary number：表示信号星座图的点数，M 为一个偶数。

Input type：表示输入由整数组成还是由比特组组成。如果本项设为 Bit，那么参数“M-ary number”必须为 2^K，K 为正整数。此时模块的输入信号是一个长度为 K 的二进制向量，且有 $K=\log_2 M$。如果本项为 Integer，那么模块接收范围在[0,M-1]的整数输入。输入可以是标量，也可以是基于帧的列向量。

Constellation ordering：星座图编码方式。如果该项设为 Binary，MATLAB 把输入的 K 个二进制符号当作一个自然二进制序列；如果该项设为 Gray，MATLAB 把输入的 K 个二

进制符号当作一个 Gray 码。

Constellation mapping：本项只有当“Constellation ordering”项设定为 User-defined 时有效。本项可以是大小为 M 的行或列向量。其中向量的第一个元素对应图中 0+ Phase offset 角，后面的元素按照逆时针旋转，最后一个元素对应星座图的点-pi/M + Phase offset。

Phase offset：表示信号星座图中的零点相位。

Output data type：设定输出数据类型。可以为 double、ingle、Fxed-point、ser-defined 或者 herit via back propagation 等。

Output word length：设定固定点数据类型的字长。本项只有当“Output data type”设定为 Fxed-point 时有效。

User-defined data type：自定义 d built-in 或 igned fixed-point 数据类型。

Set output fraction length to：设定固定点输出比例。本项只有当“Output data type”设为 Fixed-point 或 User-defined 项为固定点数据类型时可见并有效。

Output fraction length：设定固定点输出数据的分数位数。

5.2.3.2 数字相位解调

对应 M-PSK Modulator Baseband 模块，MATLAB 提供了 M-PSK Demodulator Baseband 模块，用于基带 M 元相移键控调制的解调。输入为基带形式的已调信号。模块的输入和输出都是离散的时间信号。输入可以是标量也可以是基于帧的列向量。参数“M-ary number”表示信号星座图的点数。

M-PSK Modulator Baseband 模块及其参数设定框如图 5-17 所示。

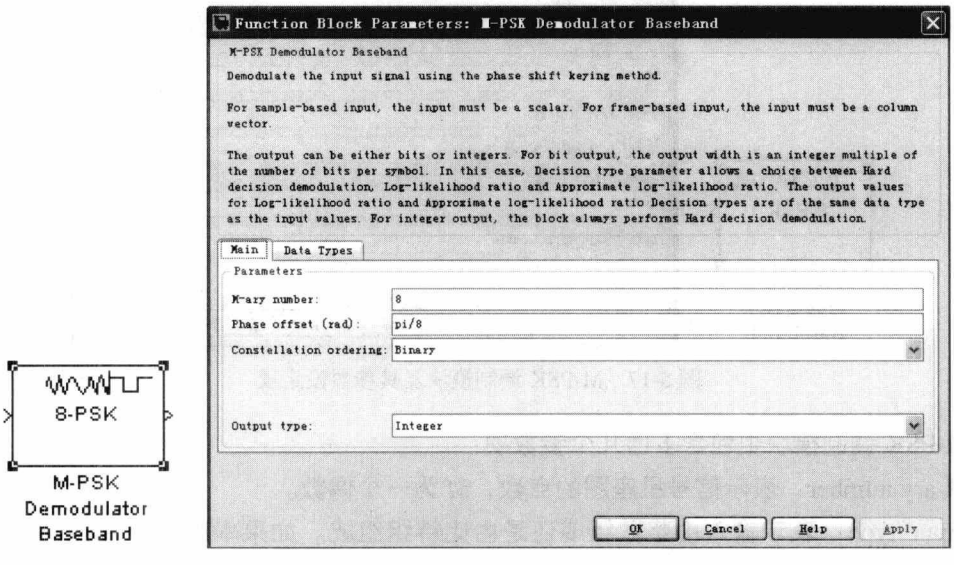


图 5-17 M-PSK 解调模块及其参数设定框

如图 5-17 所示，M-PSK 解调模块参数设定框中包含“Main”和“Data Types”两类，默认为“Main”类，如图 5-11 所示。下面分别对各类中的参数项做简单介绍。

1．“Main”类参数项说明

M-ary number：表示信号星座图的点数，M 为偶数。

Phase offset：表示信号星座图中零点的相位。

Constellation ordering：星座图编码方式，决定模块如何将符号映射成输出比特或整数。

Constellation mapping：本项只有当“Constellation ordering”项设定为 User-defined 时有效。本项可以是大小为 M 的行向量或列向量。其中向量的第一个元素对应图中 0 度角，后面的元素按照逆时针旋转，最后一个元素对应星座图的点-pi/M。

Output type：决定输出是由整数还是由比特组组成。如果本项选为 Bit，那么参数项“M-ary number”必须为 2^K，K 为正整数。

Decision type：当“Output type”选为 Bit 时出现本项，用于设定输出为 bitwise hard decision、LLR 或 approximate LLR 形式。

Noise variance source：只有当“Decision type”选定为 Approximate log-likelihood ratio 或者 Log-likelihood ratio 时显示本项。如果选择 Dialog，则在“Noise variance”中输入噪声变化。如果选择 Port，模块中显示用于设定噪声变化的端口。

Noise variance：当“Noise variance source”设定为 Dialog 时显示本项，用于设定噪声变化。

2．“Data Types”类参数项说明

“Data Types”类参数项如图 5-18 所示。

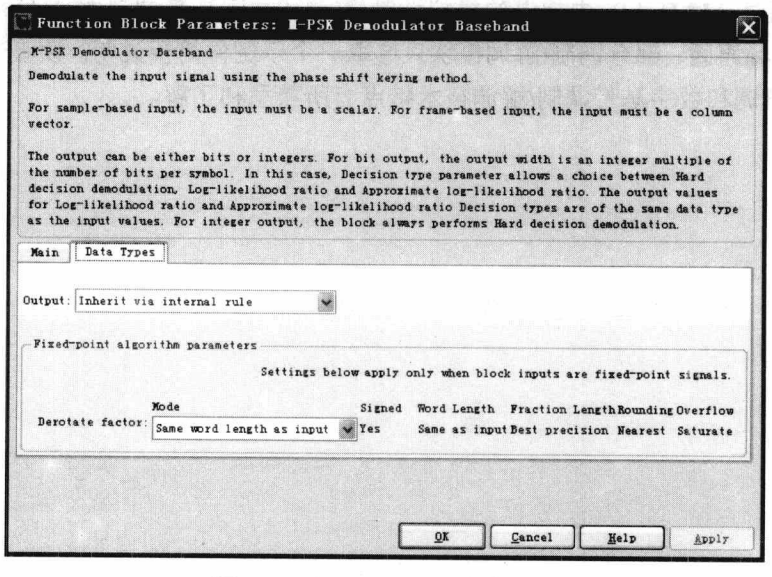


图 5-18　“Data Types”类参数项

Output：设定输出。对于比特输出，当“Decision type”设置为 Hard decision 时，输出数据类型可以为 Inherit via internal rule、Smallest unsigned integer、double、single、int8、uint8、int16、uint16、int32、uint32、boolean 等类型。对于整数输出，输出数据类型可以

是 Inherit via internal rule、Smallest unsigned integer、double、single、int8、uint8、int16、uint16、int32、uint32 类型。

如果本项设定为 Inherit via internal rule（默认项），那么数据的输出类型由输入端决定。如果输入端的输入为 floating-point type 型数据，则输出数据类型相同。如果本项设定为 fixed-point，那么输出数据类型将会和本项设定为 Smallest unsigned integer 时相同。如果本项设定为 Smallest unsigned integer，那么输出数据的类型由模型中结构参数对话框中的“Hardware Implementation”项决定。如果“Hardware Implementation”项选为 ASIC/FPGA，并且“Output type”为 Bit，那么输出数据类型为 ideal minimum one-bit size。如果“Hardware Implementation”项选为 ASIC/FPGA，并且“Output type”为 Integer，那么输出数据类型为 ideal minimum ize。

Derotate factor：本项只使用于“M-ary number”项设为 2、4、8，输入为 fixed-point 类型同时“Phase offset”项为非平凡（即该项当 M=2 时为π/2 的整数倍；当 M=4 时为π/4 的奇数倍；当 M=8 时为任意值）的情况。本项有两个可选项：Same word length as input 和 Specify word length。选定后出现设定框。在输出为比特的情况下如果“Decision type”设定为 Log-likelihood ratio 或者 Approximate log-likelihood ratio 类型时，输出与输入的数据类型相同。

5.3 本章小结

本章分别对 MATLAB 中提供的模拟调制/解调和数字基带调制/解调分别做了介绍和说明。出于篇幅原因，部分调制/解调模块只选取一个示例作简要说明。读者通过学习，将对模拟调制/解调和数字基带调制/解调技术特点有所熟悉和了解。

第 6 章　均衡器与射频损耗

在通信系统中，信号在信道间传播以及接收过程当中，存在着失真、损耗等，这些因素会对整个系统带来不利的影响。

现代通信的快速发展要求提高数据的传输率，但数据传输的速率通常会被信道失真所限制，这会引起码间干扰。当数据率低于 2400bit/s 时，码间干扰相对较小，然而当数据率高于 2400bit/s 时，就需要在调制解调器中用均衡器来校正信道失真。MATLAB 中提供了多种均衡器模块，本章中对其中的一些做简单介绍，包括 CMA 均衡器、LMS 均衡器、RLS 均衡器等。

在信号的传输和接收过程中，还存在各种各样的损耗，MATALB 也提供了各种损耗的仿真模块，本章也会做简单介绍。

6.1 CMA 均衡器

CMA 均衡器模块利用线性均衡器和恒模算法（CMA）通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用 CMA 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

在模块使用中，先要将均衡器权重初始化为一个非零向量。一般 CMA 被用于差异调制。另外，初始权重是非常重要的，一般中心抽头对应初始权重为 1，其他情况下对应的初始权重为 0。

在模块中，标记为 Input 的端口接收你想要均衡的信号，可以是标量或基于帧的向量。标记为 Equalized 的端口输出均衡后的结果。

CMA 均衡器模块及其参数设定框如图 6-1 所示。

CMA 均衡器模块包含下面几个参数项：

Number of taps：均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：信号星座图项，输入一个复向量设定调制星座图。

Step size：设定 CMA 的步长。

Leakage factor：CMA 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Initial weights：设定抽头的初始权重。

Output error：选定本项后，模块输出 $y(R\text{-}|y|^2)$形式的误差信号，其中 y 是均衡后的信

号，R 是与信号星座图相关的常数。

Output weights：选定本项后，模块输出当前权重。

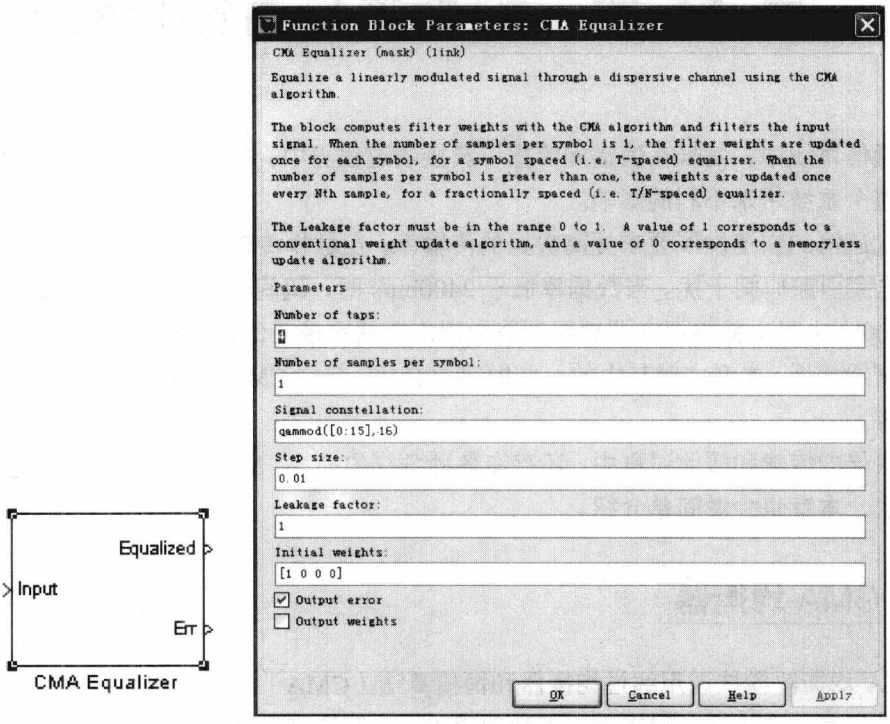


图 6-1　CMA 均衡器模块及其参数设定框

6.2 LMS 均衡器

6.2.1 LMS 判决反馈均衡器

LMS 判决反馈均衡器模块利用判决反馈均衡器和 LMS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用 LMS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定前向滤波器中心抽头的参考抽头。

LMS 判决反馈均衡器模块及其参数设定框如图 6-2 所示。

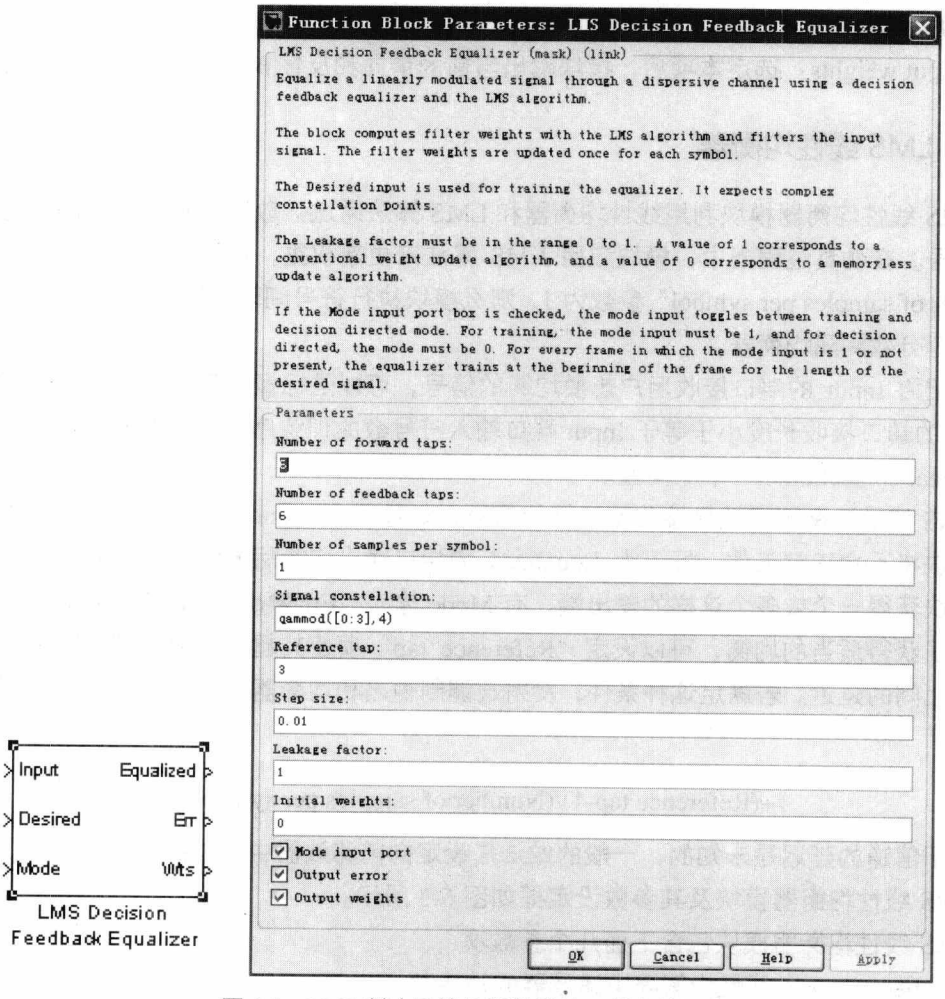


图 6-2　LMS 判决反馈均衡器模块及其参数设定框

LMS 判决反馈均衡器模块包含下面几个参数项：

Number of forward taps：判决反馈均衡器中前向滤波器的抽头数。

Number of feedback taps：判决反馈均衡器中反馈滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：信号星座图项，输入一个复向量设定调制的星座图。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Step size：设定 LMS 的步长。

Leakage factor：LMS 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Initial weights：连接前向和反馈抽头间初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信

号之间的差异。

Output weights：选定本项后，模块输出当前的前向和反馈权重。

6.2.2 LMS 线性均衡器

LMS 线性均衡器模块利用线性均衡器和 LMS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用 LMS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那些“Signal constellation”中的向量。

LMS 线性均衡器模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总延迟可以表示为：

1+(Reference tap-1)/(Number of samples per symbol)

由于信道的延迟是未知的，一般的做法是设定向前滤波器中心抽头的参考抽头。

LMS 线性均衡器模块及其参数设定框如图 6-3 所示。

LMS 线性均衡器模块包含下面几个参数项：

Number of taps：线性均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：为一个复向量，设定已调制信号星座图，由模块中的调制器决定。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Step size：设定 LMS 的步长。

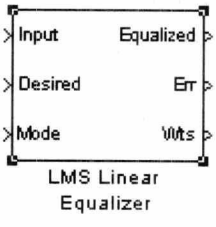
Leakage factor：LMS 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Initial weights：设定抽头初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的权重。



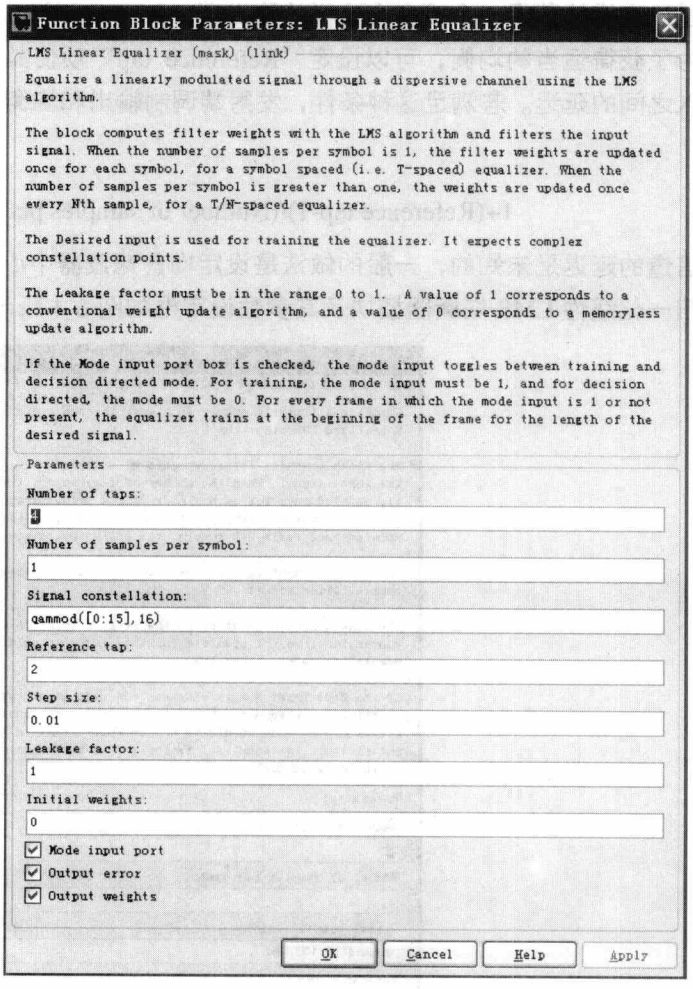


图 6-3　LMS 线性均衡器模块及其参数设定框

6.2.3　归一化 LMS 均衡器

MATLAB 中提供了两种归一化 LMS 均衡器：归一化线性 LMS 均衡器和归一化判决反馈 LMS 均衡器。这里我们对归一化线性 LMS 均衡器模块做简单说明。

归一化线性 LMS 均衡器模块利用线性均衡器和归一化的 LMS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用归一化的 LMS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那些“Signal constellation”中的向量。

归一化线性 LMS 均衡器模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可

以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定向前滤波器中心抽头的参考抽头。

归一化线性 LMS 均衡器模块及其参数设定框如图 6-4 所示。

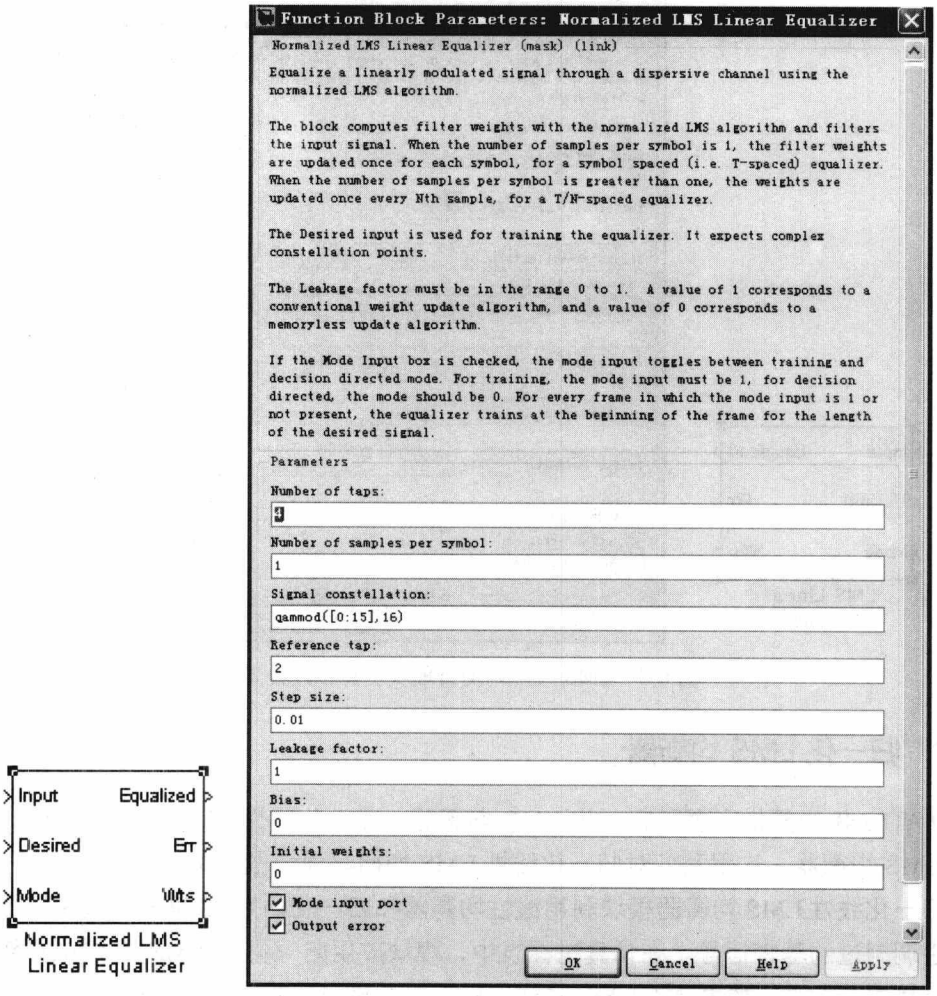


图 6-4　归一化线性 LMS 均衡器模块及其参数设定框

归一化线性 LMS 均衡器模块包含下面几个参数项：

Number of taps：线性均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation；为一个复向量，设定已调制信号星座图，由模块中的调制器决定。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Step size：设定 LMS 的步长。

Leakage factor：LMS 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Bias：归一化 LMS 算法中的 Bias 参数，为一非负实数。本项用于克服当输入信号很小时算法可能遇到的困难。

Initial weights：设定抽头初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的权重。

6.2.4　符号 LMS 均衡器

MATLAB 中提供了两种符号 LMS 均衡器：符号线性 LMS 均衡器和符号判决反馈 LMS 均衡器。这里对符号线性 LMS 均衡器模块做简单说明。

符号线性 LMS 均衡器模块利用线性均衡器和有符号的 LMS 算法通过弥散信道来补偿线性调制过的基带信号。对应于“Update algorithm”项的模块的算法，有下面几种：Sign LMS、Sign Regressor LMS、Sign Sign LMS。

在仿真过程当中，对每个符号模块应用特殊的有符号 LMS 算法来更新权重。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那些“Signal constellation”中的向量。

符号线性 LMS 均衡器模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总的延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定前向滤波器中心抽头的参考抽头。

符号线性 LMS 均衡器模块及其参数设定框如图 6-5 所示。

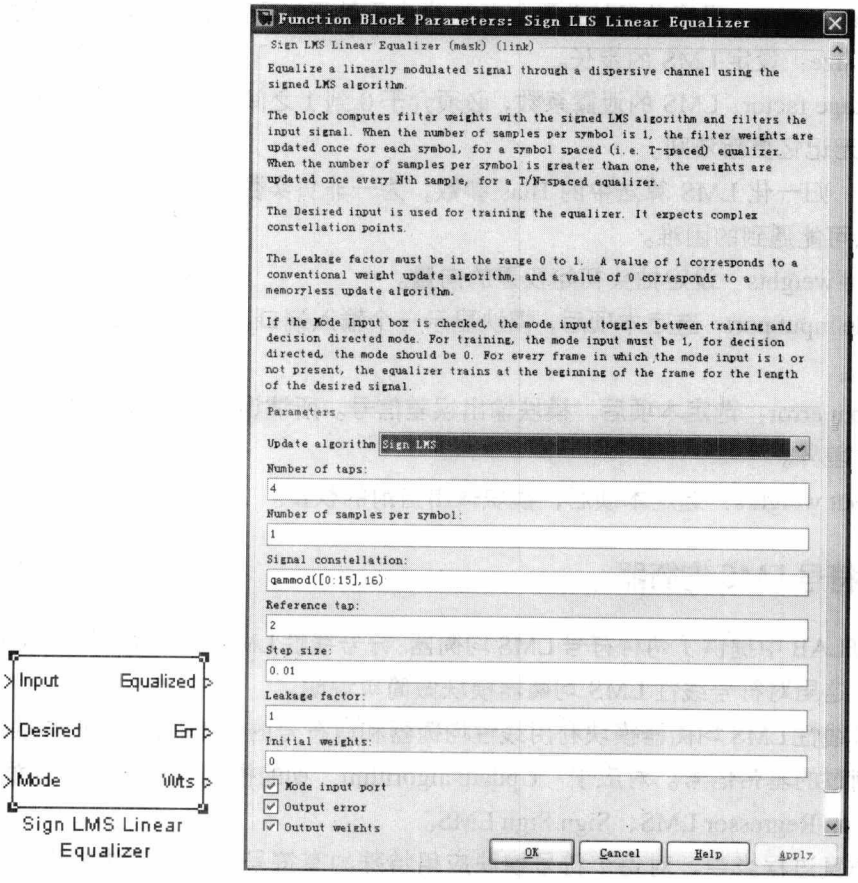


图 6-5　符号线性 LMS 均衡器模块及其参数设定框

符号线性 LMS 均衡器模块包含下面几个参数项：

Update algorithm：设定模块用于更新均衡器权重的有符号 LMS 算法的特定类型。

Number of taps：线性均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：为一个复向量，设定已调制信号星座图，由模块中的调制器决定。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Step size：设定有符号 LMS 算法的步长。

Leakage factor：LMS 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Initial weights：设定抽头初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的权重。

6.2.5　变步长 LMS 均衡器

MATLAB 中提供了两种变步长 LMS 均衡器：变步长线性 LMS 均衡器和变步长判决反馈 LMS 均衡器。这里我们以变步长线性 LMS 均衡器为例做简单说明。

变步长线性 LMS 均衡器模块利用线性均衡器和变步长的 LMS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用变步长的 LMS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那些“Signal constellation”中的向量。

变步长线性 LMS 均衡器模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定向前滤波器中心抽头的参考抽头。

变步长线性 LMS 均衡器模块及其参数设定框如图 6-6 所示。

变步长线性 LMS 均衡器模块包含下面几个参数项：

Number of taps：线性均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：为一个复向量，设定已调制信号星座图，由模块中的调制器决定。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Initial step size：设定变步长 LMS 算法在仿真初始阶段的初始步长。

Increment step size：设定步长增量。

Minimum step size：设定可取步长的最小值。

Maximum step size：设定可取步长的最大值。

Leakage factor：LMS 的泄露系数，必须介于 0 到 1 之间。1 对应常规的权重更新法则；0 对应于无记忆更新法则。

Initial weights：设定抽头初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的权重。

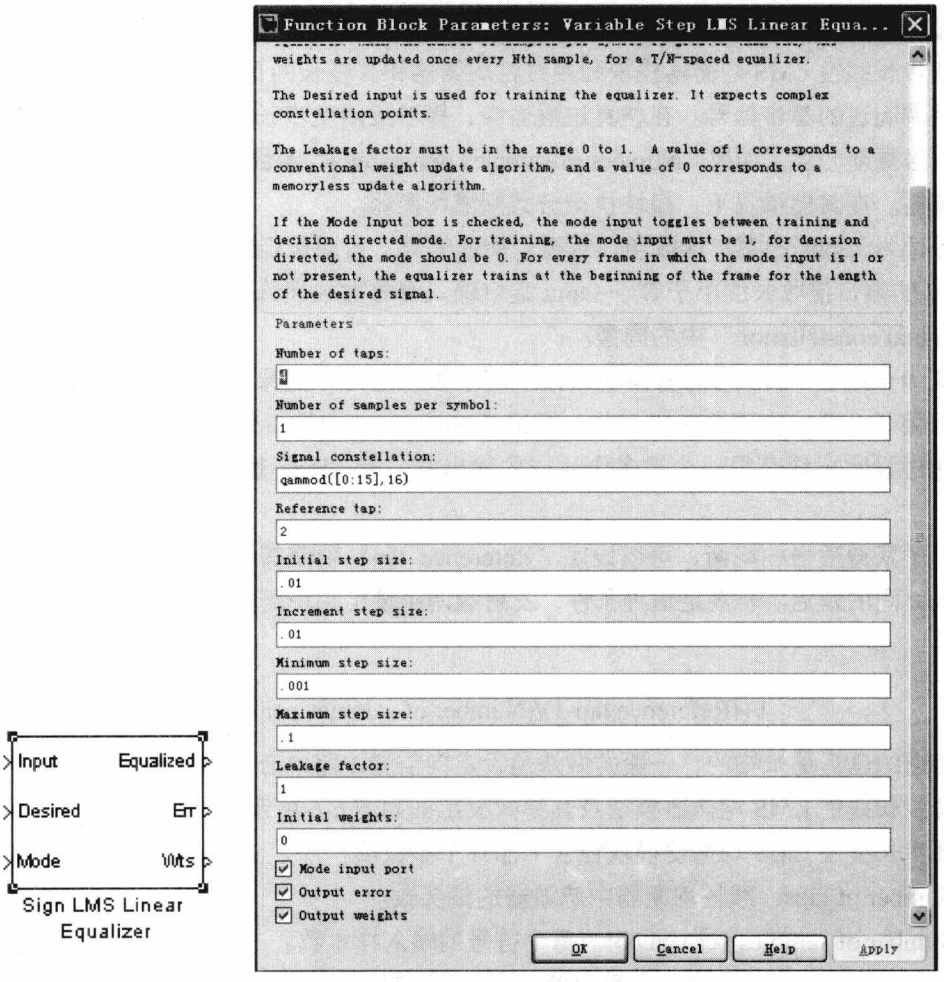


图 6-6　变步长线性 LMS 均衡器模块及其参数设定框

6.3 RLS 均衡器

6.3.1 RLS 判决反馈均衡器

RLS 判决反馈均衡器模块利用判决反馈均衡器和 RLS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用 RLS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那

些“Signal constellation”中的向量。

模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间总的延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定向前滤波器中心抽头的参考抽头。

RLS 判决反馈均衡器模块及其参数设定框如图 6-7 所示。

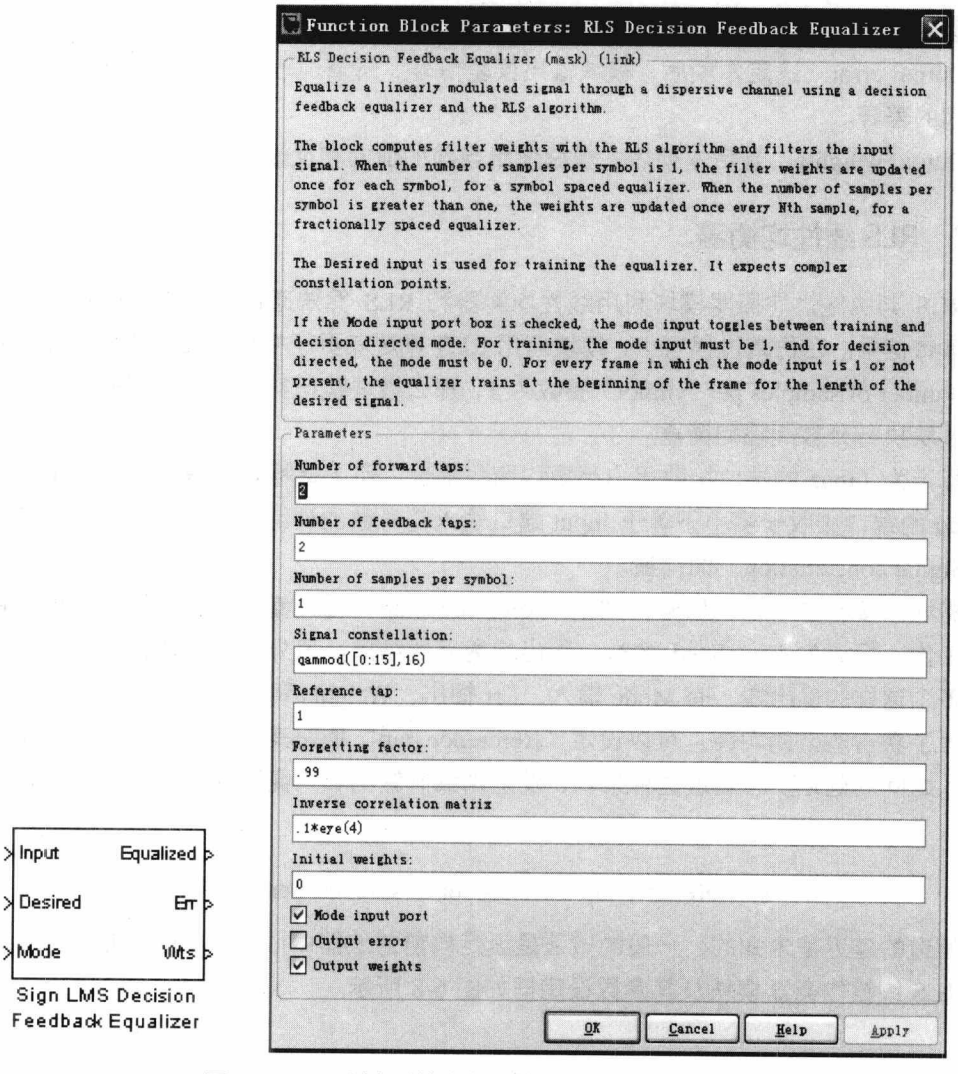


图 6-7　RLS 判决反馈均衡器模块及其参数设定框

RLS 判决反馈均衡器模块包含下面几个参数项：

Number of forward taps：判决反馈均衡器中向前滤波器的抽头数。

Number of feedback taps：判决反馈均衡器中反馈滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：信号星座图项，输入一个复向量设定调制星座图。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Forgetting factor：设定 RLS 算法中的遗忘因子，必须在 0～1 之间。

Inverse correlation matrix：设定逆相关矩阵的初始值，矩阵必须为 $N \times N$ 阶方阵。其中 N 为前向和反馈的抽头的总数。

Initial weights：设定向前和反馈抽头间初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的前向和反馈权重。

6.3.2 RLS 线性均衡器

RLS 判决线性均衡器模块利用线性均衡器和 RLS 算法通过弥散信道来补偿线性调制过的基带信号。在仿真过程当中，模块应用 RLS 算法来更新权重，每个符号更新一次。如果“Number of samples per symbol”参数为 1，那么模块执行符号间隔均衡器。在其他情况下，模块执行分数间隔均衡器。

标记为 Input 的端口接收用户想要均衡的信号，可以是标量或基于帧的向量。标记为 Desired 的端口接收长度小于等于 Input 端口输入符号数的训练序列。有效的训练符号是那些“Signal constellation”的向量。

模块只接收基于帧的信号，如果“Reference tap”项的值大于等于帧长，模块将不会正常工作。标记为 Equalized 的端口输出均衡后的结果。另外也可以通过设定模块获得一个或多个这样的输出端，有 Mode 输入、Err 输出、Weights 输出等。

为了获得适当的均衡，可以设定“Reference tap”项使其超过发射端调制输出和均衡器输入之间的延迟。若满足这种条件，发射端调制输出和均衡器输出之间的总延迟可以表示为：

$$1+(\text{Reference tap}-1)/(\text{Number of samples per symbol})$$

由于信道的延迟是未知的，一般的做法是设定向前滤波器中心抽头的参考抽头。

RLS 线性均衡器模块及其参数设定框如图 6-8 所示。

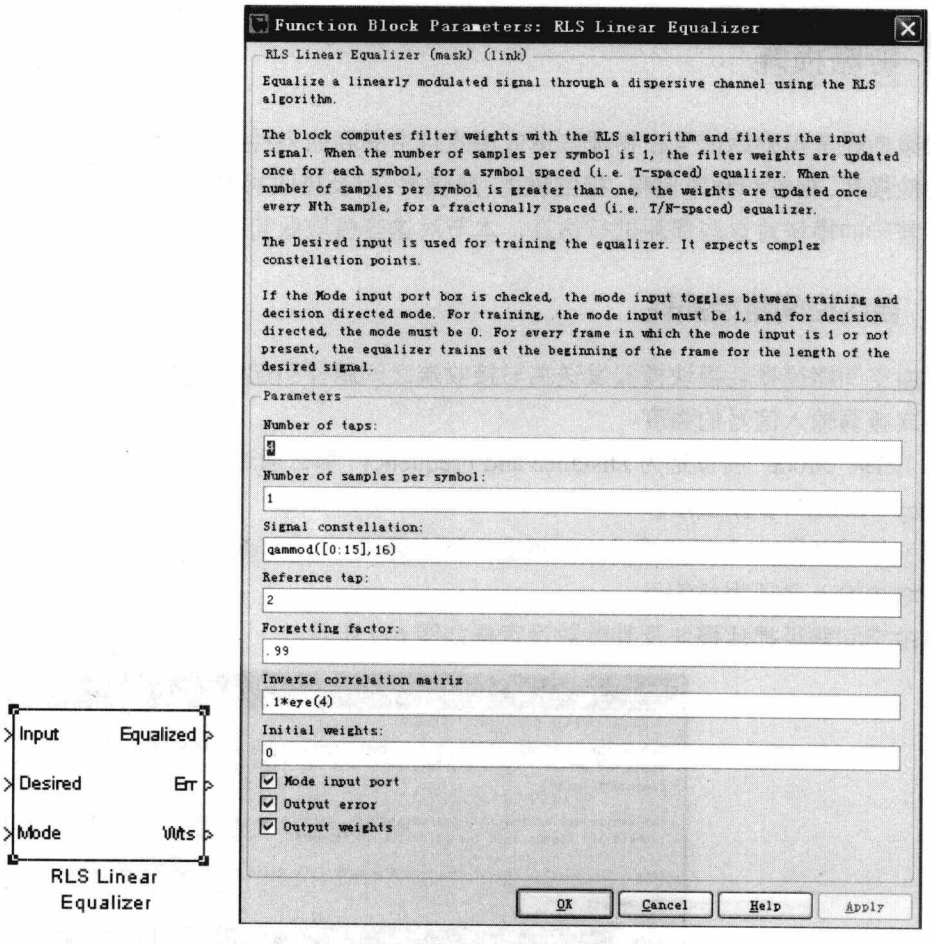


图 6-8　RLS 线性均衡器模块及其参数设定框

RLS 线性均衡器模块包含下面几个参数项：

Number of taps：线性均衡器中滤波器的抽头数。

Number of samples per symbol：每个符号的输入样本数。

Signal constellation：为一个复向量，设定已调制信号星座图，由模块中的调制器决定。

Reference tap：设定参考抽头数，为一个小于等于均衡器中抽头数的正整数。

Forgetting factor：设定 RLS 算法中的遗忘因子，必须在 0～1 之间。

Inverse correlation matrix：设定逆相关矩阵的初始值，矩阵必须为 $N \times N$ 阶方阵。其中 N 为前向和反馈的抽头的总数。

Initial weights：设定抽头初始权重的向量。

Mode input port：选定本项后，模块显示一个输入端口，有 training 和 decision-directed 两种模式。

Output error：选定本项后，模块输出误差信号。所谓误差信号就是均衡信号和参考信号之间的差异。

Output weights：选定本项后，模块输出当前的权重。

6.4 射频损耗

射频损耗是指射频信号在物理信道或接收机中受到的各种损耗，包括信号在自由空间中的传输损耗、相位和频率偏移、相位噪声、热噪声，以及接收机的非线性作用等。MATLAB提供了若干的模块对这些损耗进行仿真，本节对这些模块做简要介绍。

6.4.1 自由空间路径损耗

自由空间路径损耗模块模拟发送端与接收端之间路径引起的信号损耗。模块通过下面两种方式减弱输入信号的强度：

（1）如果“Mode”项设定为 Distance and Frequency，那么通过“Distance (km)”和“Carrier frequency (MHz)”参数设定。

（2）如果“Mode”项设定为 Decibels，那么通过“Loss (dB)”参数设定。

模块的输入必须为复信号。

自由空间路径损耗模块及其参数设定框如图 6-9 所示。

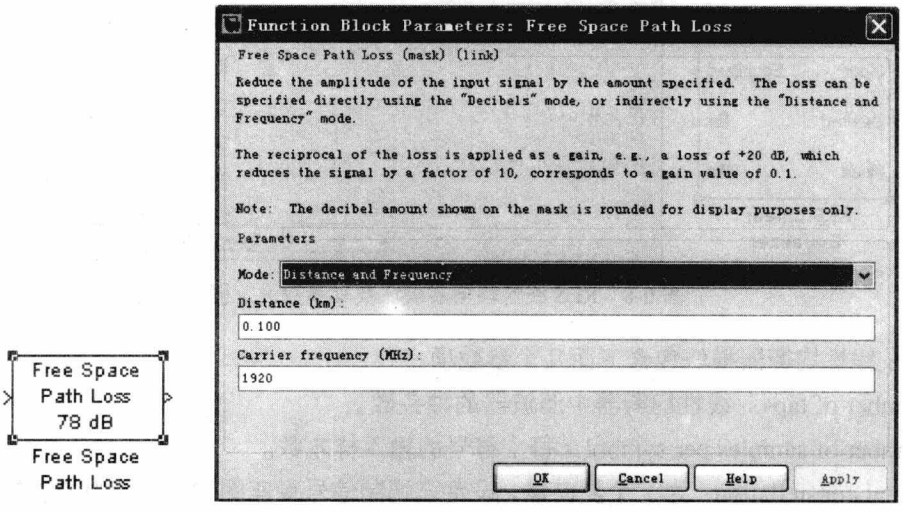


图 6-9 自由空间路径损耗模块及其参数设定框

自由空间路径损耗模块包含下面几个参数项：

Mode：设定自由空间损耗模块的工作模式。有 Decibels 和 Distance and Frequency 两种模式。

Loss (dB)：设定自由空间损耗模块的路径损耗。只有在“Mode”设定为 Decibels 时本项有效。

Distance(km)：设定发送端与接收端之间距离。只有在“Mode”设定为 Distance and Frequency 时本项有效。

Carrier frequency (MHz)：设定输入信号的载波频率。只有在“Mode”设定为 Distance

and Frequency 时本项有效。

6.4.2　相位噪声

相位噪声模块的输入信号是复数形式的基带信号。模块首先通过 AWGN 模块产生一个加性高斯白噪声信号，然后将这个加性高斯噪声信号作为相位噪声叠加到输入的信号中。右击模块，选定 Look under mask 项，模块将会显示相位噪声的内部结构图，如图 6-10 所示。

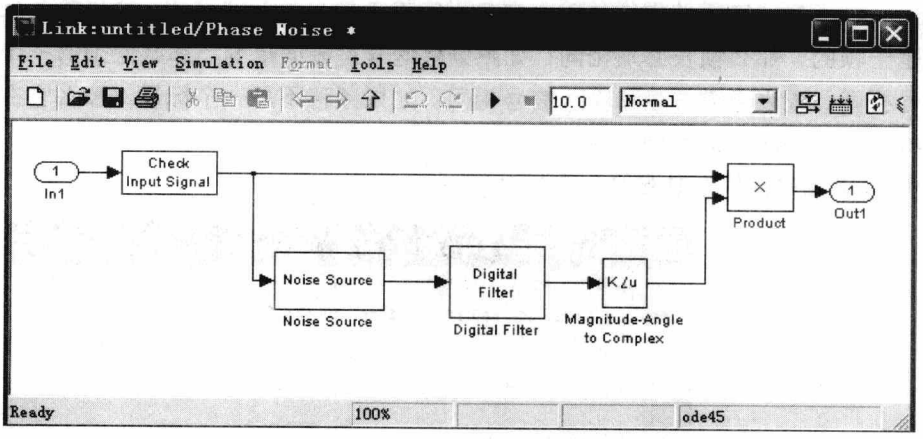


图 6-10　相位噪声的内部结构图

相位噪声模块及其参数设定框如图 6-11 所示。

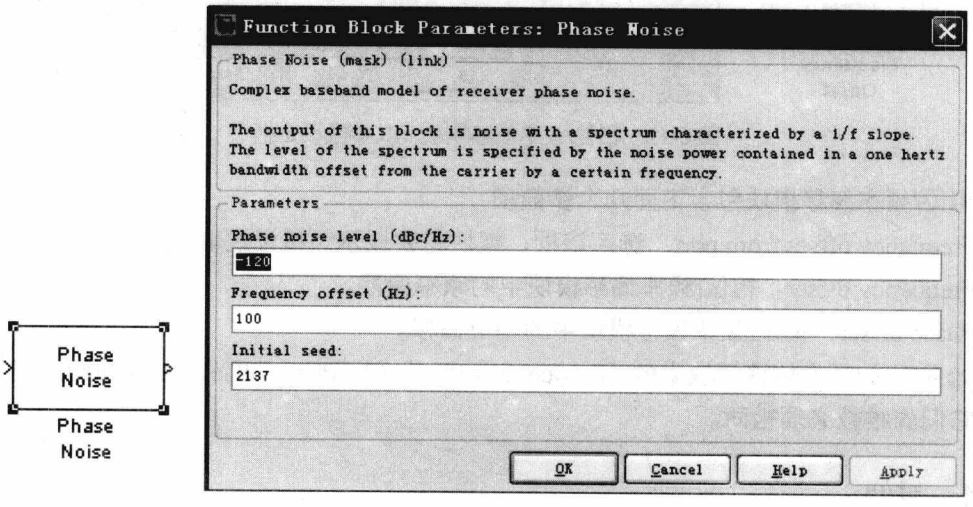


图 6-11　相位噪声模块及其参数设定框

相位噪声模块包含下面几个参数项：

Phase noise level (dBc/Hz)：模块产生的相位噪声的强度。

Frequency offset (Hz)：模块产生的频率偏移。

Initial seed：模块的初始化种子，必须为非负整数。

6.4.3 相位/频率偏移

相位/频率偏移模块用于产生输入信号的相位/频率偏移。模块中不会出现任何的延时。模块的输入可以是类型为 double 或 single 的实数或复数基带信号，输出的数据类型和输入相同。

模块通过“Phase offset”项设定输入信号的相位偏移。

模块通过“Frequency offset”项设定输入信号的频率偏移，另外也可选定“Frequency offset from port”项，在模块提供的输入端口设定输入信号的频率偏移。如果输入和频移信号均是基于帧的，那么帧长必须相同；如果频移信号是多通道的，那么必须有一个通道和输入信号相同，或者通道总数和输入信号相同；如果频移不是一个标量，那么它在通道中的总采样数必须和输入信号相对应。

相位/频率偏移模块及其参数设定框如图 6-12 所示。

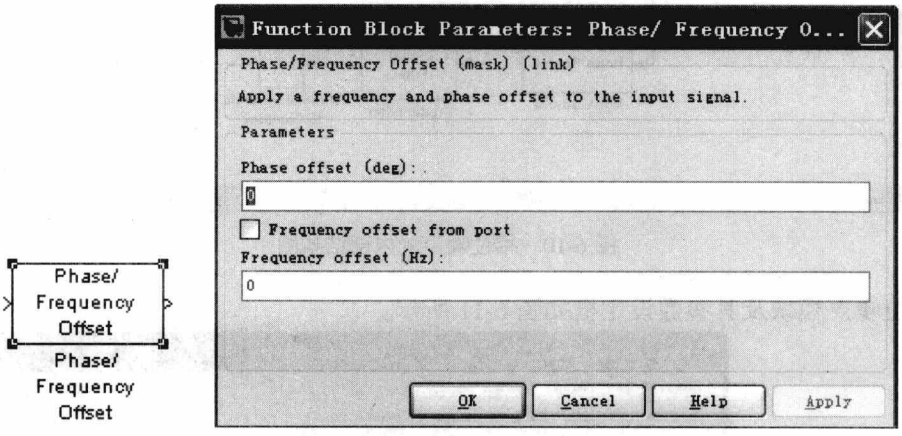


图 6-12 相位/频率偏移模块及其参数设定框

相位/频率偏移模块包含下面几个参数项：

Frequency offset from port：选定该项，模块将会显示频移设定输入端。

Frequency offset：相位/频率偏移模块中的频率偏移。

Phase offset：相位/频率偏移模块中的相位偏移。

如果相位/频率偏移模块中的“Frequency offset”和“Phase offset”项均为向量或矩阵，那么它们的维数必须相同。

6.4.4 其他

除了前面讲到的三个模块之外，MATLAB 还提供了 I/Q Imbalance 模块、Memoryless Nonlinearity 模块和 Receiver Thermal Noise 模块等，本节简要地介绍一下这些模块。

1. I/Q Imbalance 模块

I/Q Imbalance 模块是对信号的 I/Q 支路失衡的模拟。I/Q 支路失衡一般是由传输通道

对 I 支路和 Q 支路信号的不同损耗引起的。模块的输入信号是复数形式的基带信号，这个信号的 I 支路分量和 Q 支路分量在不同的幅度、相位以及直流分量的影响下，改变原有的数值。

2．Memoryless Nonlinearity 模块

Memoryless Nonlinearity（无记忆非线性）模块对输入的基带复信号进行无记忆非线性处理，这种处理过程一般对无线通信系统中的接收机的射频损耗进行仿真。模块中共有五种处理方式：Cubic polynomial、Hyperbolic tangent、Saleh model、Ghorbani model 和 Rapp model。

3．Receiver Thermal Noise 模块

Receiver Thermal Noise 模块用于模拟复数形式的基带信号的热噪声效应。热噪声是由于导体内部自由电子的布朗运动引起的噪声。导体中的每一个自由电子都具有一定的热能，它们处在无规则的热运动之中，并且和其他粒子之间不停地发生碰撞，呈现出随机的和曲折的布朗运动。所有的布朗运动形成通过导体的电流，此时电流的方向是随机的，因此平均值为 0。同时，电子的这种随机运动还会产生一个交流电流的成分，这个交流成分就是热噪声。

6.5　本章小结

本章主要介绍了 MATLAB 中提供的各种均衡器模块以及射频损耗模块。对 CMA 均衡器、LMS 据衡器、RLS 均衡器等多个模块做了详细介绍，同时对射频损耗仿真模块做了简要说明。

第 7 章　通信滤波器

在通信系统中，很多地方都需要用到滤波器，如调制解调、波形成型等。滤波器是一种十分重要的线性时不变系统，它是一个能让某些频率的分量通过，而完全拒绝其他频率成分的系统。但这是一种理想化的滤波器，实际上很难实现完全拒绝其他频率成分。因为滤波器是一个因果系统，在实际中不能实现完全拒绝某些频率分量，只能使其衰减到一定的指标。上面所说的滤波器是狭义上的，广义上讲，任何能够对某些频率进行修正的系统都可以看成滤波器。

滤波器可以分成模拟滤波器和数字滤波器两种。目前的应用中数字滤波器相对比较广泛。MATLAB 中提供了许多滤波模块，本章对其中的一些做简要的介绍：包括数字滤波器和模拟滤波器的设计和分析，以及常用滤波器模块的介绍等。

7.1 滤波器设计模块

MATLAB 中提供了供用户自行设计、分析、实现滤波器的滤波器设计模块。既包含数字滤波器设计，也提供模拟滤波器设计。本节分别对它们做详细阐述。

7.1.1 数字滤波器设计

MATLAB 中提供了 Digital Filter Design 模块实现 FIR 和 IIR 数字滤波器。Digital Filter Design 模块通过 Filter Design and Analysis Tool（FDATool）图形用户界面实现用户设计的 FIR 和 IIR 数字滤波器。本模块可实现与 Digital Filter block 相同的滤波器。

Digital Filter Design 模块将指定的滤波器应用到每个通道的离散时间输入信号上，输出滤波结果。该结果在数值上与 Digital Filter block、MATLAB 中的 filter 函数，以及滤波器设计工具箱中的 filter 函数所得结果相同。

模块的输入可以是基于帧或基于采样的向量、矩阵。在模块中，基于帧的向量或矩阵中的每一列，或基于采样的向量或矩阵中的每一个元素，都被看成一个信道。模块对每一个信道实行单独滤波。输出和输入具有相同的维数和状态。

Digital Filter Design 模块如图 7-1 所示。

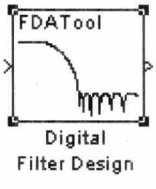


图 7-1　Digital Filter Design 模块

这里通过一个低通 FIR 数字滤波器的设计为例，说明 Digital Filter Design 模块的使用。设计可按照下面几个步骤进行：

① 建立一个新的仿真窗口。

② 打开 Signal Processing Blockset Filtering 库，找到 Filter Designs 库，将其中的 Digital Filter Design 模块拖到仿真窗口中。

③ 双击 Digital Filter Design 模块，打开模块图形用户界面。

④ 在图形用户界面设定参数：Response Type=Lowpass；Design Method=FIR，Equiripple；Filter Order=Minimum order；Units=Normalized（0 to 1）；wpass=0.2；wstop=0.5。

⑤ 单击图形用户界面中的 Design Filter 按钮，确定参数设定。

⑥ 打开 Edit 菜单，选定 Convert Structure 项，打开 Convert Structure 对话框。

⑦ 在对话框中选定 Direct-Form FIR Transposed，单击 OK 按钮。

⑧ 为模块设定的低通滤波器重新命名。

Digital Filter Design 模块图形用户界面如图 7-2 所示。

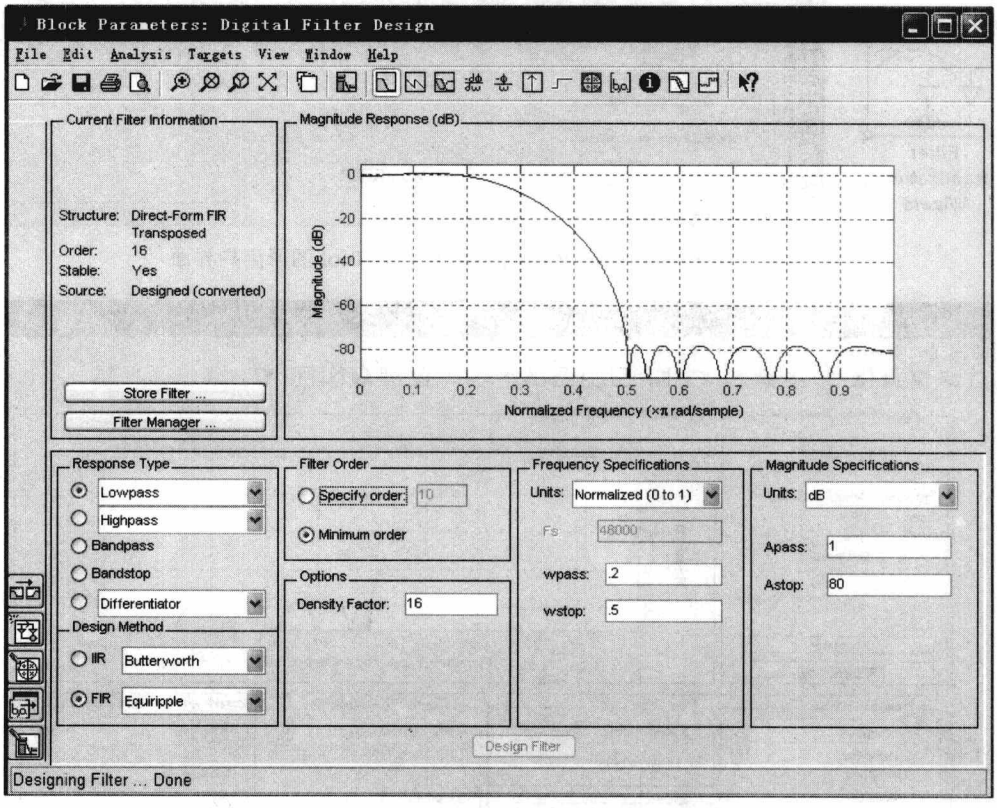


图 7-2　Digital Filter Design 模块图形用户界面

实际上 Digital Filter Design 模块是利用 FDATool 图形用户界面进行滤波器设计的，Filter Realization Wizard 模块和 FDATool 图形用户界面如图 7-3 所示。

单击 FDATool 图形用户界面左端的 图标，出现参数设定界面，按照上文中低通滤波器的参数设定本界面中的参数，可以获得相同的结果，如图 7-4 所示。

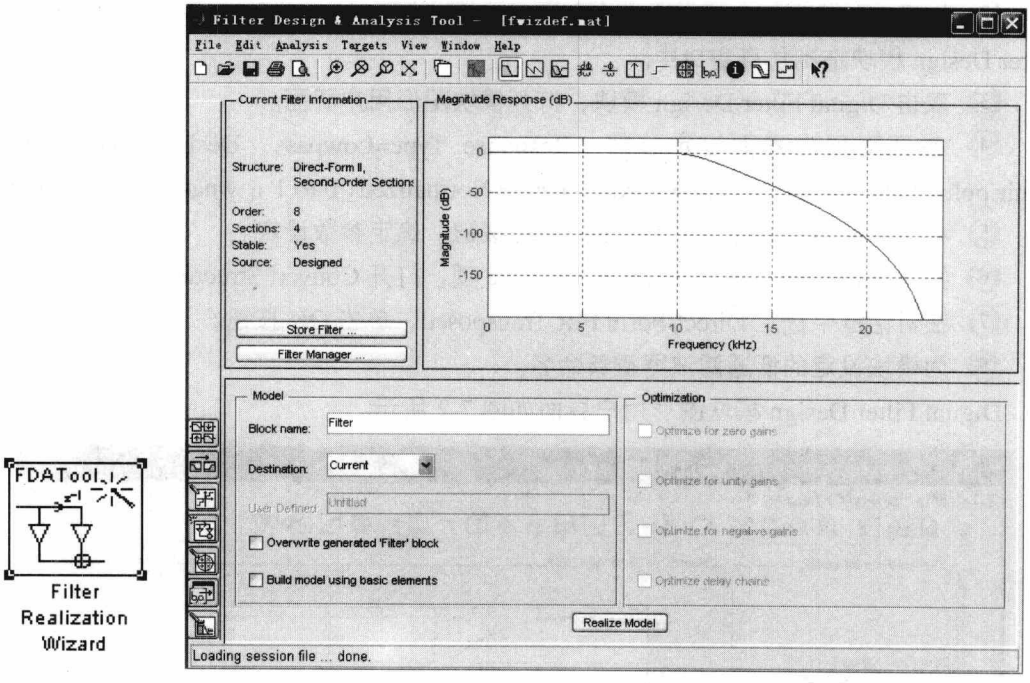


图 7-3　Filter Realization Wizard 模块和 FDATool 图形用户界面

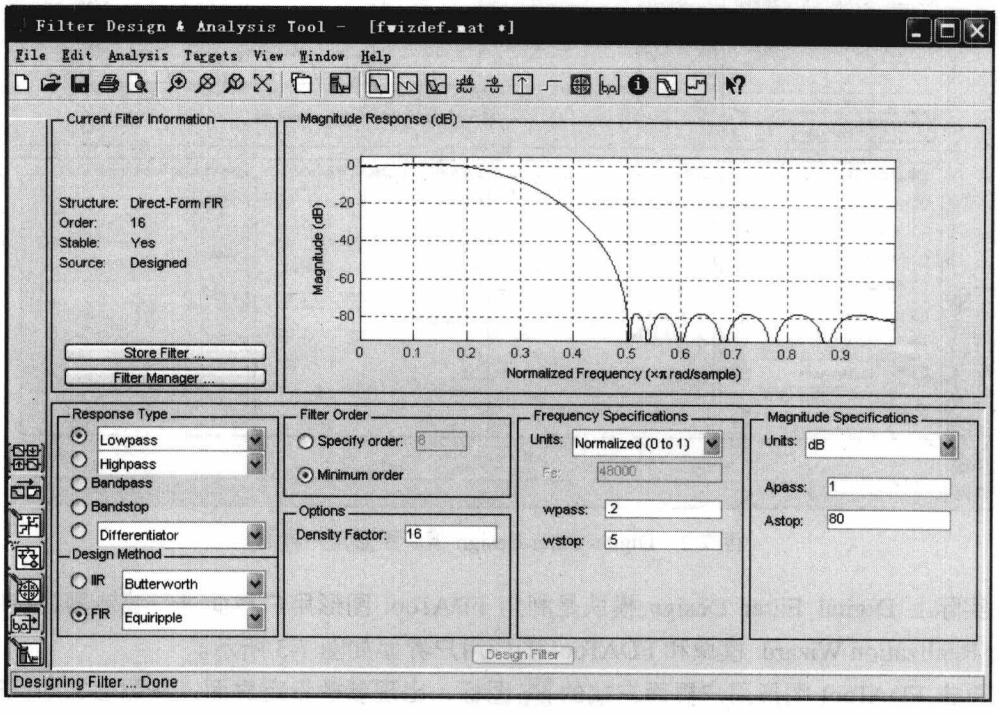


图 7-4　Filter Realization Wizard 设定的低通滤波器

由图 7-2 和图 7-4 可知，两者的结果相同。因此，利用 Digital Filter Design 和 Filter Realization Wizard 模块中的任意一个，均可以设计滤波器。两个模块具有若干相似性：

（1）两个模块都提供了 FDATool 图形用户界面来进行滤波器的设计和分析。

（2）在 Double-precision 中，两个模块的输出在数字上都与 Filter Design Toolbox 中的 filter 函数和 Signal Processing Toolbox 中的 filter 输出匹配。

除了上面的相似点之外，这两个模块还有一些差别：

（1）两个模块都支持 Single-precision 和 Double-precision 的浮点运算，但 Filter Realization Wizard 还支持定点运算。

（2）Digital Filter Design 实现了非常高效的滤波器，在仿真设计和 C 代码产生时在速度和存储空间上采用了优化措施。而 Filter Realization Wizard 用 sum、gain 和 Unit Delay block 实现滤波器。和 Digital Filter Design 实现的滤波器相比，Filter Realization Wizard 产生的滤波器效率要低很多。

（3）两个模块都支持多种相同的基本滤波器，但是 Filter Realization Wizard 支持的结构更多一些。

（4）Digital Filter Design 模块能够对多通道的信号滤波，而 Filter Realization Wizard 模块只能实现单通道滤波。

综合 Digital Filter Design 模块和 Filter Realization Wizard 模块的异同，可以发现：

1. Digital Filter Design 模块适用于下面几种设计情况：

（1）用于仿真的 Single-precision 或 Double-precision 滤波器。

（2）多通道信号滤波。

（3）产生在嵌入式系统中应用的高度优化的 ANSIC 代码。

2. Filter Realization Wizard 模块适用于下面几种设计情况：

（1）用于仿真在 DSP 芯片、field-programmable gate array（FPGA）或 Application-Specific Integrated Circuit（ASIC）中定点滤波器。

（2）用于仿真 Digital Filter Design 模块不支持的 Single-precision 或 Double-precision 滤波器。

（3）用于滤波器结构的可视化。

（4）用于快速生成多个滤波器。

7.1.2　模拟滤波器设计

除了数字滤波器设计模块，MATLAB 还提供了一个模拟滤波器设计模块 Analog Filter Design，本节对此模块做简单介绍。

Analog Filter Design 模块能够设计并实现 Butterworth、Chebyshev type I、Chebyshev type II，或者 elliptic 类型的低通、高通、带通或带阻滤波器。

Analog Filter Design 模块的输入必须为基于采样的连续实值标量信号。

Analog Filter Design 模块及其参数设定框如图 7-5 所示。

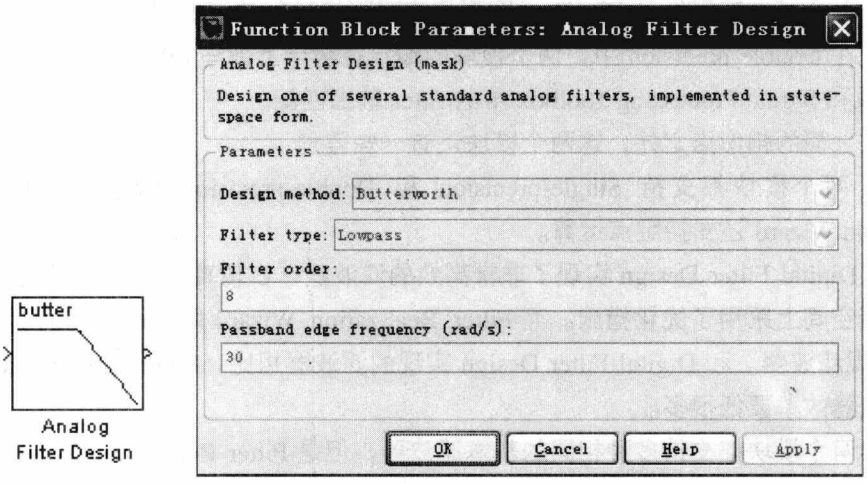


图 7-5　Analog Filter Design 模块及其参数设定框

Analog Filter Design 模块中包含下面几个参数项：

Design method：滤波器的设计方法，有 Butterworth、Chebyshev type I、Chebyshev type II，或者 elliptic 等类型。

Filter type：设定滤波器的类型，包括 Lowpass、Highpass、Bandpass 或 Bandstop。

Filter order：设定滤波器阶数。对于低通和高通滤波器，阶数等于本项参数值；对于带通和带阻滤波器，阶数为本项参数值的两倍。

Passband edge frequency：设定 Butterworth、Chebyshev type I、elliptic 类型高通、低通滤波器的通带边缘频率。

Lower passband edge frequency：设定 Butterworth、Chebyshev type I、elliptic 类型带通、带阻滤波器的上边缘频率。

Upper passband edge frequency：设定 Butterworth、Chebyshev type I、elliptic 类型带通、带阻滤波器的下边缘频率。

Stopband edge frequency：设定 Chebyshev type II 类型高通、低通滤波器的阻带边缘频率。

Lower stopband edge frequency：设定 Chebyshev type II 类型带通、带阻滤波器的较低阻带边缘频率。

Upper stopband edge frequency：设定 Chebyshev type II 类型带通、带阻滤波器的较高阻带边缘频率。

Passband ripple in dB：设定 Chebyshev type I 及 elliptic 类型滤波器的通带波纹。

Stopband attenuation in dB：设定 Chebyshev type II 及 elliptic 类型滤波器的阻带内的衰减。

7.2 理想矩形脉冲滤波器

除了数字和模拟滤波器设计模块之外，MATLAB 还提供了一些常用的滤波器模块。本节我们简单介绍一下理想矩形脉冲滤波器模块。

理想矩形脉冲滤波器模块利用矩形脉冲对输入信号提高采样频率或成形。模块将每个输入采样复制 N 次，N 为模块中的“Pulse length”参数项的值。对输入采样复制之后，模块还可以归一化输出信号或应用线性幅值增益。

如果模块中“Pulse delay”项非零，那么在开始复制输入值之前，模块输出零点个数。模块的输入可以是标量或者基于帧的列向量，并支持 double、single 和 fixed-point 等数据类型。如果输入基于采样，那么输出采样实际是输入采样时间的 $1/N$。输出与输入的维数相同。此时模块的“Input sampling mode”项必须设为 Sample-based。如果输入是基于帧的 $K\times 1$ 阶矩阵，那么输出是基于帧的 $K\times N\times 1$ 阶矩阵。输出帧周期与输入帧周期对应。此时模块的“Input sampling mode”项必须设为 Frame-based。

模块的归一化可以通过“Normalize output signal”和“Linear amplitude gain”两个参数项来设定：

在默认情况下，“Normalize output signal”项是选定的。如果撤销选定，那么“Normalization method”项消失。模块将会用“Linear amplitude gain”项参数乘以复制的值。

如果“Normalize output signal”项选定，那么模块将会显示“Normalization method”项。模块将会缩放复制值从而满足下面两个条件中的一个：

（1）每个脉冲的采样总数等于模块复制的初始输入值。

（2）每个脉冲的能量等于模块复制的初始输入值，也就是每个脉冲中矩形采样的和等于输入值的平方。

模块应用“Normalization method”项的缩放设定之后，将会用缩放后的信号乘以“Linear amplitude gain”参数项的值。

理想矩形脉冲滤波器模块及其参数设定框如图 7-6 所示。

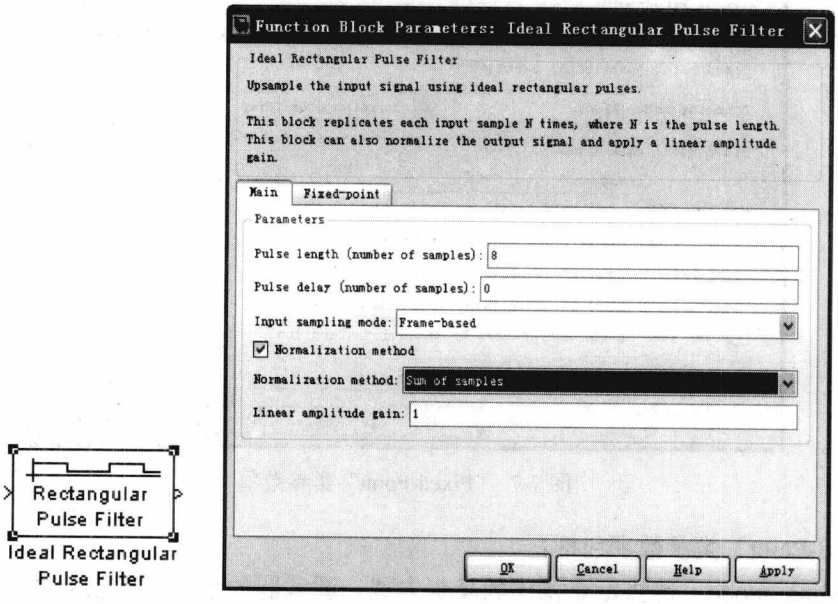


图 7-6　理想矩形脉冲滤波器模块及其参数设定框

理想矩形脉冲滤波器模块包含两大类参数项："Main"类和"Fixed-Point"类。这里分别做简要介绍。

1."Main"类参数项

"Main"类参数如图 7-6 所示。它包含下面几个参数项：

Pulse length：用于设定每个输出脉冲中的采样数，就是当模块生成输出信号时对每个输入值的复制次数。

Pulse delay：脉冲延迟项。表示仿真初始阶段，在开始复制输入值之前，模块输出零点的个数。

Input sampling mode：设定输入信号的类型，有 Frame-based 和 Sample-based 两种类型。

Normalize output signal：选定本项之后，在应用线性幅值增益之前，模块将会对复制值进行缩放。

Normalization method：模块缩放复制值的数量。

Linear amplitude gain：用于对输出信号的缩放，需要为正的标量。

2."Fixed-Point"类参数项

理想矩形脉冲滤波器模块中还包含"Fixed-Point"类参数项，如图 7-7 所示。

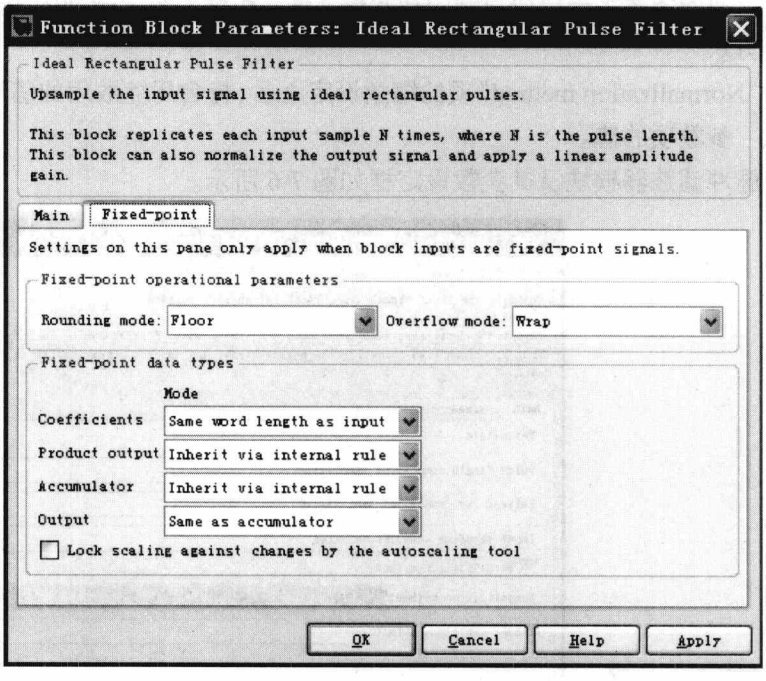


图 7-7 "Fixed-Point"类参数项

"Fixed-Point"类参数项包括：

Rounding mode：选择定点操作的凑整方式。滤波器的系数并不服从本参数，它们通常为 Nearest 型凑整。

Overflow mode：选择定点操作的溢出方式。滤波器的系数并不服从本参数，它们通常

是饱和的。

Coefficients：选择如何设定滤波器系数的字长和小数长度。当选择 Same word length as input 时，滤波器系数的字长和模块的输入相对应。小数长度自动设置为 binary-point；当选择 Specify word length 时，可以自行输入系数的字长，单位是比特。小数长度自动设置为 binary-point；当选择 Binary point scaling 时，可以自行输入字长与小数长度，单位是比特。此时，可以单独输入分子系数与分母系数的小数长度；当选择 Slope and bias scaling 时，可以自行输入滤波器系数的字长和斜率。可以单独输入分子系数与分母系数的斜率。本模块要求斜率为 2 的幂次方，偏置为 0。

Product output：本参数用于指定用户如何设置乘积输出字长和小数长度。当本项选定为 Same as input 时，乘积输出字长和小数长度与模块的输入相对应；当本项选定为 Binary point scaling 时，可以自行设定乘积输出的字长和小数长度；当本项选定为 Slope and bias scaling 时，可以自行设定乘积输出的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

Accumulator：本参数用于指定用户如何设置累加器的字长和小数长度。当本项选定为 Same as input 时，累加器字长和小数长度与模块的输入相对应；当本项选定为 Same as product output 时，累加器字长和小数长度与模块的输出相对应；当本项选定为 Binary point scaling 时，可以自行设定累加器的字长和小数长度；当本项选定为 Slope and bias scaling 时，可以自行设定累加器的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

Output：选择如何设定输出字长和小数长度。当本项选定为 Same as input 时，输出字长和小数长度与输入相对应；当本项选定为 Same as accumulator 时，输出字长和小数长度与累加器的字长和小数长度相对应；当本项选定为 Binary point scaling 时，可以自行设定输出的字长和小数长度；当本项选为 Slope and bias scaling 时，可以自行设定输出的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

7.3 升余弦滤波器

MATLAB 中还提供了升余弦滤波器模块，这其中又包含升余弦发射滤波器模块和升余弦接收滤波器模块。本节分别对它们做简单介绍。

7.3.1 升余弦发射滤波器

升余弦发射滤波器模块利用常规升余弦 FIR 滤波器或平方根升余弦 FIR 滤波器对输入信号提高采样频率或成形。

如果滚降系数为 R，符号周期为 T，那么常规升余弦滤波器的脉冲响应可以表示为：

$$h(t)=\frac{\sin(\pi t/T)}{(\pi t/T)}\cdot\frac{\cos(\pi Rt/T)}{\left(1-4R^2t^2/T^2\right)} \qquad (式\ 7\text{-}1)$$

而平方根升余弦滤波器的脉冲响应可以表示为：

$$h(t)=4R\frac{\cos\left((1+R)\pi t/T\right)+\dfrac{\sin\left((1-R)\pi t/T\right)}{(4Rt/T)}}{\pi\sqrt{T}\left(1-(4Rt/T)^2\right)} \qquad (式\ 7\text{-}2)$$

模块中的“Group delay”参数是滤波器响应起始点与峰值之间的符号周期数。本项和模块中的提高采样频率参数 N 决定了滤波器的脉冲响应为 2*N* Group delay+1。

模块中的“Rolloff factor”参数是滤波器的滚降系数。必须为 0 到 1 之间的实数。本项决定滤波器的超出带宽。例如当本项为 0.5 时，表示滤波器的带宽是输入采样频率的 1.5 倍。

模块中的“Filter gain”项显示模块如何归一化滤波器参数。

（1）如果本项选为 Normalized，那么模块将会应用自动缩放。

当“Filter type”是 Normal 时，模块归一化滤波器参数使得峰值参数等于 1；当“Filter type”是 Square root 时，模块归一化滤波器，使得滤波器与本身的卷积生成一个峰值参数为 1 的常规升余弦滤波器。

（2）如果本项选为 User-specified，那么滤波器的带宽增益为：

常规滤波器：$20\log_{10}\left(\text{Upsampling factor}(N)\times\text{Linear amplitude filter gain}\right)$。

平方根滤波器：$20\log_{10}\left(\sqrt{\text{Upsampling factor}(N)}\times\text{Linear amplitude filter gain}\right)$。

模块的输入信号必须是标量或基于帧的列向量。模块支持 double、single、fixed-point 等数据类型。参数项“Input sampling mode”决定模块的输入是基于帧还是基于采样。本项和“Upsampling factor”参数项 N 共同决定输出信号特征。

如果输入是基于采样的标量，那么输出也是基于采样的标量，且输出采样时间是输入采样时间的 N 倍。

如果输入是基于帧的，那么输出也是基于帧的向量，且向量长度是输入向量长度的 N 倍。输出帧与输入帧的周期相同。

升余弦发射滤波器模块及其参数设定框如图 7-8 所示。

升余弦发射滤波器模块包含两大类参数项：“Main”类和“Fixed-Point”类。这里分别做简要介绍。

1.“Main”类参数项

“Main”类参数如图 7-8 所示。它包含下面几个参数项：

Filter type：设定升余弦滤波器的类型，有 Square root 和 Normal 两种。

Group delay：滤波器响应起始点与峰值之间的符号周期数，必须为一正整数。

Rolloff factor：滤波器的滚降系数，为 0 到 1 之间的实数。

Input sampling mode：设定输入采样类型，有 Frame-based 和 Sample-based 两种。

Upsampling factor：提高采样频率系数。表示滤波后的输出信号中每个符号的采样数，必须为大于 1 的整数。

Filter gain：滤波器增益项，决定模块如何缩放滤波器参数。有 Normalized 和 User-specified 两种方式。

Linear amplitude filter gain：线性振幅增益项，为一用于缩放滤波器参数的正的标量。本项只有当“Filter gain”项选定为 User-specified 时出现。

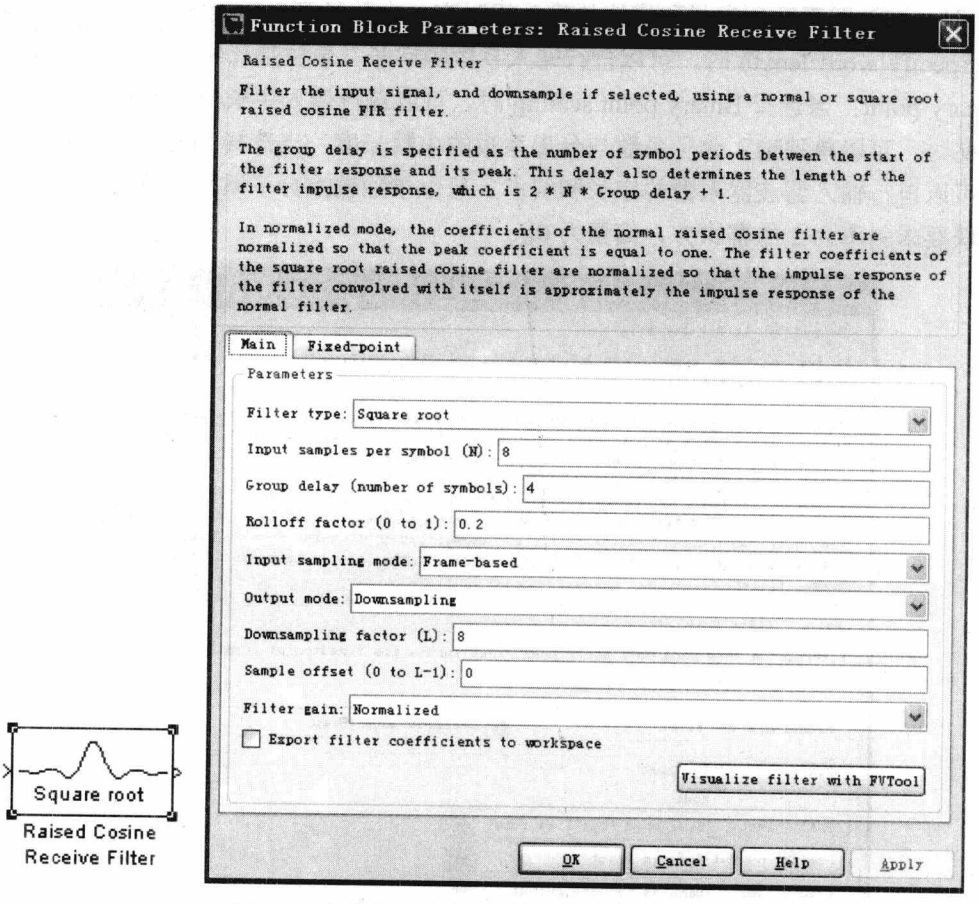


图 7-8　升余弦发射滤波器模块及其参数设定框

Export filter coefficients to workspace：滤波器参数输出到工作空间项，选定本项后，模块将在 MATLAB 中创造一个包含滤波器参数的变量。

Coefficient variable name：用于设定模块在 MATLAB 工作空间中创造的变量名。本项只有当“Export filter coefficients to workspace”选定时显示。

Visualize filter with FVTool：本项为一按钮。单击本按钮后，MATLAB 将会启动滤波器可视化工具 fvtool，模块的参数发生任何变化时将会对升余弦滤波器进行分析。

2.“Fixed-Point”类参数项

升余弦发射滤波器模块中还包含“Fixed-Point”类参数项，如图 7-9 所示。

“Fixed-Point”类参数项包括下面几个参数：

Rounding mode：选择定点操作的凑整方式。滤波器的系数并不服从本参数，它们通常为 Nearest 型凑整。

Overflow mode：选择定点操作的溢出方式。滤波器的系数并不服从本参数，它们通常

是饱和的。

Coefficients：选择如何设定滤波器系数的字长和小数长度。当选择 Same word length as input 时，滤波器系数的字长和模块的输入相对应。小数长度自动设置为 binary-point；当选择 Specify word length 时，可以自行输入系数的字长，单位是比特。小数长度自动设置为 binary-point；当选择 Binary point scaling 时，可以自行输入字长与小数长度，单位是比特。此时，可以单独输入分子系数与分母系数的小数长度；当选择 Slope and bias scaling 时，可以自行输入滤波器系数的字长和斜率。可以单独输入分子系数与分母系数的斜率。本模块要求斜率为 2 的幂次方，偏置为 0。

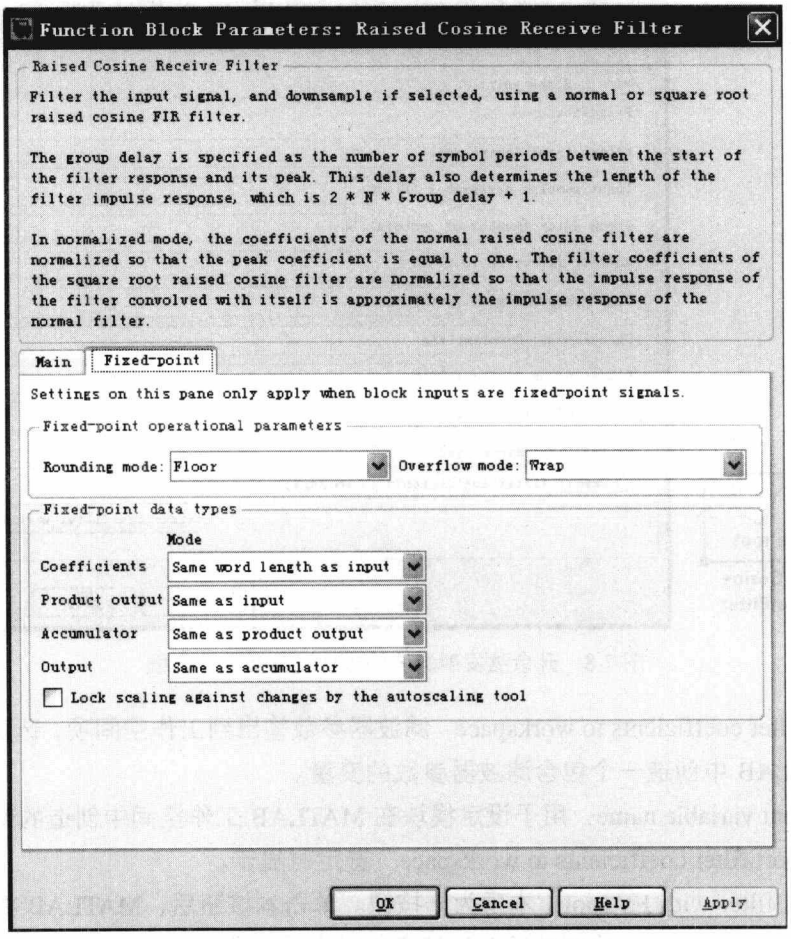


图 7-9 “Fixed-Point”类参数项

Product output：本参数用于指定用户如何设置乘积输出字长和小数长度。当本项选定为 Same as input 时，乘积输出字长和小数长度与模块的输入相对应；当本项选定为 Binary point scaling 时，可以自行设定乘积输出的字长和小数长度；当本项选定为 Slope and bias scaling 时，可以自行设定乘积输出的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

Accumulator：本参数用于指定用户如何设置累加器的字长和小数长度。当本项选定为 Same as input 时，累加器字长和小数长度与模块的输入相对应；当本项选定为 Same as

product output 时，累加器字长和小数长度与模块的输出相对应；当本项选定为 Binary point scaling 时，可以自行设定累加器的字长和小数长度；当本项选定为 Slope and bias scaling 时，可以自行设定累加器的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

Output：选择如何设定输出字长和小数长度。当本项选定为 Same as input 时，输出字长和小数长度与输入相对应；当本项选定为 Same as accumulator 时，输出字长和小数长度与累加器的字长和小数长度相对应；当本项选定为 Binary point scaling 时，可以自行设定输出的字长和小数长度；当本项选为 Slope and bias scaling 时，可以自行设定输出的字长和斜率。此时要求斜率为 2 的幂次方，偏置为 0。

7.3.2　升余弦接收滤波器

升余弦接收滤波器模块利用常规升余弦 FIR 滤波器或平方根升余弦 FIR 滤波器对过滤输入信号。如果“Output mode”项设定为 Downsampling，它也会减小滤波后的信号采样频率。

当“Output mode”项设定为 Downsampling 且“Downsampling factor”项参数为 L 时，模块将按照下面的方法保留采样的 $1/L$。

如果“Sample offset”项为 0，模块选择滤波后信号序列为 1，L+1，2*L+1，3*L+1…等的采样。

如果“Sample offset”项为小于 L 的正整数，那么模块去掉初始的“Sample offset”项正整数个采样，再按照上面的方法来降低采样频率。

模块的输入信号必须是标量或基于帧的列向量。模块支持 double、single、fixed-point 等数据类型。

如果“Output mode”项设为 0，那么输入和输出信号具有相同的采样方式、采样时间、向量长度。

如果“Output mode”项设为 Downsampling，并且“Downsampling factor”项参数为 L，那么 L 和输入采样方式决定输出信号的特征。

如果输入是基于采样的标量，那么输出也是基于采样的标量，且输出采样时间是输入采样时间的 $1/L$。

如果输入是基于帧的，那么输出也是基于帧的向量，且向量长度是输入向量长度的 $1/L$。输出帧与输入帧的周期相同。

升余弦接收滤波器模块及其参数设定框如图 7-10 所示。

升余弦接收滤波器模块包含两大类参数项：“Main”类和“Fixed-Point”类。这里分别做简要介绍。

1．“Main”类参数项

“Main”类参数如图 7-10 所示。它包含下面几个参数项：

Filter type：设定升余弦滤波器的类型，有 Square root 和 Normal 两种类型。

Input samples per symbol：输入信号中每个符号的采样数，必须为一个大于 1 的整数。

Group delay：滤波器响应起始点与峰值之间的符号周期数，必须为一正整数。

Rolloff factor：滤波器的滚降系数，为 0 到 1 之间的实数。

Input sampling mode：设定输入采样类型，有 Frame-based 和 Sample-based 两种类型。

Output mode：决定模块是否对过滤后的信号降低采样频率。有 Downsampling 和 None 两种。

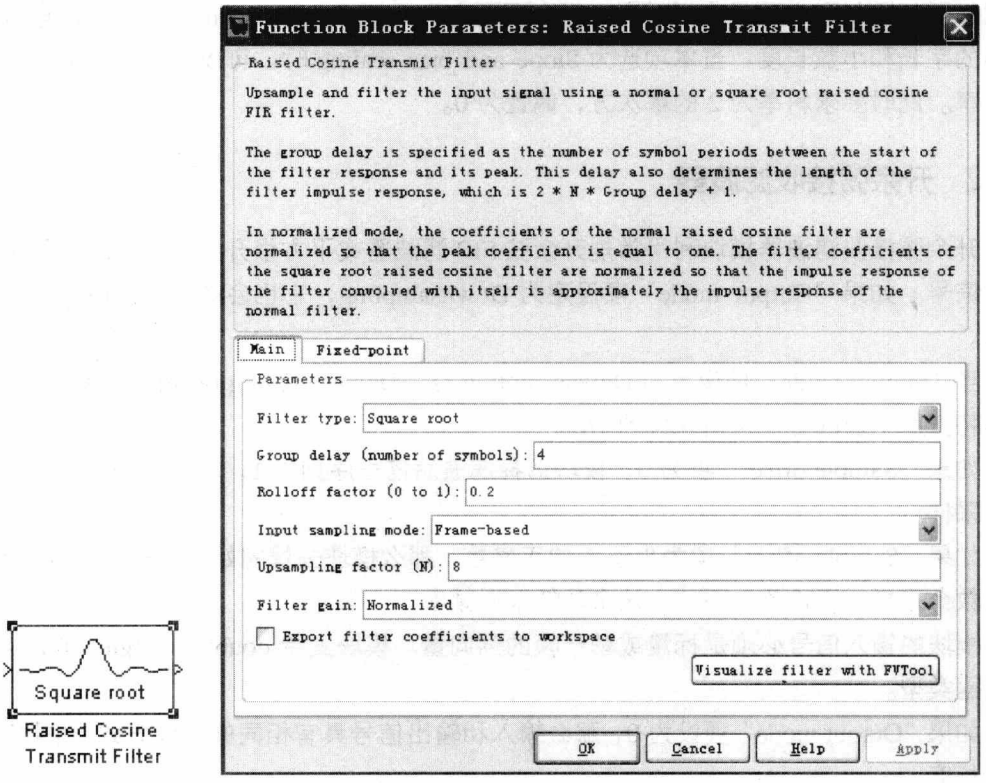


图 7-10　升余弦接收滤波器模块及其参数设定框

Downsampling factor：模块对过滤后的信号降低采样频率参数。本项只有当“Output mode”项选为 Downsampling 时显示。

Sample offset：模块降低采样频率之前丢弃的过滤后采样的总数。本项只有当“Output mode”项选为 Downsampling 时显示。

Filter gain：滤波器增益项，决定模块如何缩放滤波器参数。有 Normalized 和 User-specified 两种方式。

Linear amplitude filter gain：线性振幅增益项，为一用于缩放滤波器参数的正的标量。本项只有当“Filter gain”项选定为 User-specified 时出现。

Export filter coefficients to workspace：滤波器参数输出到工作空间项，选定本项后，模块将在 MATLAB 中创造一个包含滤波器参数的变量。

Coefficient variable name：用于设定模块在 MATLAB 工作空间中创造的变量名。本项只有当“Export filter coefficients to workspace”选定时显示。

Visualize filter with FVTool：本项为一按钮。单击本按钮后，MATLAB 将会启动滤波器可视化工具 fvtool，模块的参数发生任何变化时将会对升余弦滤波器进行分析。

2．“Fixed-Point”类参数项

升余弦接收滤波器模块中还包含“Fixed-Point”类参数项，如图 7-11 所示。

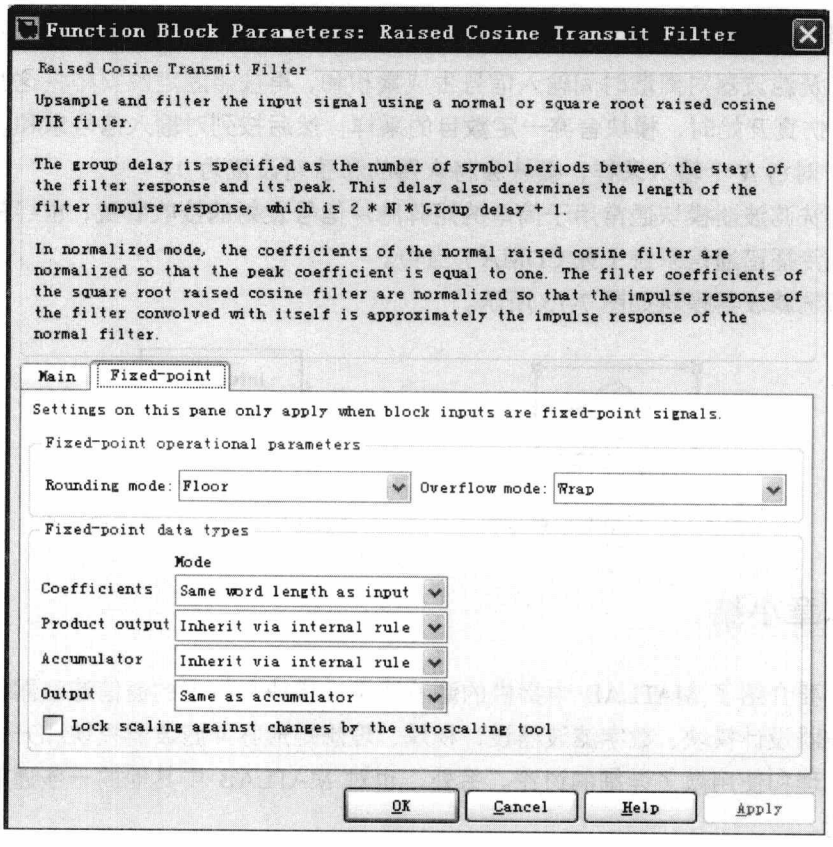


图 7-11　“Fixed-Point”类参数项

升余弦接收滤波器模块中的“Fixed-Point”类参数项和升余弦发射滤波器模块相同，这里不再赘述。

7.4　其他

MATLAB 中既提供了数字滤波器设计、模拟滤波器设计模块，又提供了很多滤波器模块，除了上面提到的理想矩形脉冲滤波器和升余弦滤波器模块之外，还包括一些其他的滤波器，本节我们分别对它们做简要的介绍，各滤波器模块的具体应用可以参考 MATLAB 的 help 部分。

1．高斯滤波器模块

高斯滤波器模块利用高斯 FIR 滤波器对输入的信号进行滤波处理。模块本身认为输入

信号是已经提高采样频率后的信号，所以每个符号的输入采样数 N 至少要是 2。模块中高斯响应为 $h(t)=\dfrac{\exp(\dfrac{-t^2}{2\delta^2})}{\sqrt{2\pi}\cdot\delta}$，其中 $\delta=\dfrac{\sqrt{\ln 2}}{2\pi BT}$，$B$ 为滤波器的 3dB 带宽。

高斯滤波器模块如图 7-12 所示。

2．积分清洗滤波器

积分清洗滤波器对离散时间输入信号生成累积和，再按照固定的安排将这些累积和重新设为 0。仿真开始时，模块舍弃一定数目的采样，然后按列对输入信号求和。当积分周期为 N 时，对每 N 个输入采样，模块会将求得的和重新设置为 0。

积分清洗滤波器模块通常用于简单的矩阵脉冲信号发射的接收装置，也经常用于光纤光学和无线传播谱通信系统（如 CDMA）当中。

积分清洗滤波器模块如图 7-13 所示。

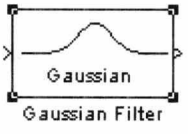


图 7-12　高斯滤波器模块

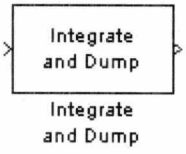


图 7-13　积分清洗滤波器模块

7.5 本章小结

本章主要介绍了 MATLAB 中提供的滤波器设计模块和特定的通信滤波器模块。分别对模拟滤波器设计模块、数字滤波器设计模块、理想矩形脉冲滤波器模块和升余弦滤波器模块等的原理和使用做了详细的讨论。另外，也对 MATLAB 中其他的一些通信滤波器做了简单介绍。

第 8 章　差错控制编码/译码

在移动通信系统中，发送端发出的无线信号肯定要受到噪声（如随机噪声、突发噪声等）的影响。如果信号的传输波形受到破坏，将会使得接收端可能发生错误判决。这就要求在通信系统中，尽可能地减小噪声干扰的影响。消除或降低噪声干扰可以采用均衡器，或提高信号的发射功率等，但是这些方法通常不能满足传输要求。

为了解决这个问题，MATLAB 提供了差错控制编码译码，它的目的就在于减少通信系统中的传输错误。在信道编码过程中，发送端通过某种方式对信息序列进行计算，得到相应的检错/纠错编码，然后把这个检错/纠错编码附加到信息序列中，经过载波调制之后发送出去。接收端对接收到的信号进行解调，从中恢复出包含原始信息序列和检错/纠错编码的二进制序列。对这个序列实施信道编码的逆过程之后，就得到了所需的信息序列，同时也知道了该序列是否存在着传输错误。

MATLAB 通信工具箱中支持常用的差错控制编码译码技术，包括线性分组码、循环卷积码。另外它和循环冗余码同属于信源编码的范畴，但为了论述的方便，把它放在本章一并讨论。本章将依次介绍这三种编码的原理和使用方法。

8.1 线性分组码

分组码是将信息分组进行编码译码，各组之间没有联系。本节讨论的都是线性分组码。分组码的编码包括两个部分：

（1）将信源的输出序列分成 K 位一组的消息组。

（2）编码器根据一定的编码规则将 K 位消息变换成 n 个码元的码字。

线性分组码是通过线性关系将 K 位消息变换成 n 个码元的码字，构成(n, K)线性分组码。对于(n,k)线性分组码，生成矩阵是一个 $K\times n$ 的矩阵。设输入的信息为 $\boldsymbol{m}=[m_1,m_2,...,m_k]$，生成的码字为 $\boldsymbol{v}=[v_1,v_2,\cdots,v_k]$，则 $\boldsymbol{V}=\boldsymbol{mG}$，其中 $\boldsymbol{G}$ 为生成矩阵。生成矩阵的各行向量为码字空间的基底，由于一个关键的基底选择不是唯一的，所以生成矩阵 G 的选择也不是唯一的。对于生成码字中前 $\boldsymbol{k}$ 位与信息完全相同的码称为系统码。这样对于系统码其生成的矩阵可以表示为 $\boldsymbol{G}=[\boldsymbol{I}_k\ \ \boldsymbol{P}]$，$\boldsymbol{I}_k$ 为 $k\times k$ 的单位矩阵，$\boldsymbol{P}$ 为一个 $k\times(n-k)$ 的单位矩阵。

由于(n, k)码的生成矩阵 $\boldsymbol{G}$ 表示的是 n 维空间的一个 k 维子空间，那么一定存在一个 $n-k$ 维的子空间与 $\boldsymbol{G}$ 表示的子空间正交，称为 $\boldsymbol{G}$ 行空间的零化空间。我们用一个 $(n-k)\times n$ 的矩阵 $\boldsymbol{H}$ 的行向量来表示这个零化空间。则有如下关系：$\boldsymbol{GH}^{\mathrm{T}}=0$ 或 $\boldsymbol{HG}^{\mathrm{T}}=0$。矩阵 H 称为(n, k)码的一致校验矩阵。

8.1.1 BCH 编码/译码

BCH 码解决了生成多项式与纠错能力的关系问题，可以方便地纠正多个随机错误，因此是一种特别重要的循环码。其定义为：二进制或 q 进制循环码生成多项式 $g(x)$，若含有以下 δ-1 个连续根：$\alpha,\alpha^2,\alpha^3,\cdots,\alpha^{\delta-1}$，则由该 $g(x)$生成的(n, k)循环码称为二进制或 q 进制 BCH 码。码的生成多项式为：

$$g(x)=\mathrm{LCM}\{m_1(x),m_2(x),\cdots,m_{\delta-1}(x)\} \qquad (式\ 8\text{-}1)$$

其中，$m_i(x)$ 为 α^i 的最小多项式。LCM 表示最小公倍数。BCH 码的最小距离为 δ。

8.1.1.1 BCH 编码模块

BCH 编码模块的输入必须是有 K 的整数倍个元素的基于帧的列向量，每 K 个输入元素代表一个信号字。模块中完成（N，K）的 BCH 编码。其中 N 具有（2^M-1）的形式，$3\leqslant M\leqslant 16$。如果 N 小于（2^M-1），那么模块认为码长减小了 2^M-1-N；如果 N 大于（2^M-1），那么必须在模块参数项“Primitive polynomial”中设定适当的 M 值。

BCH 编码模块及其参数设定框如图 8-1 所示。

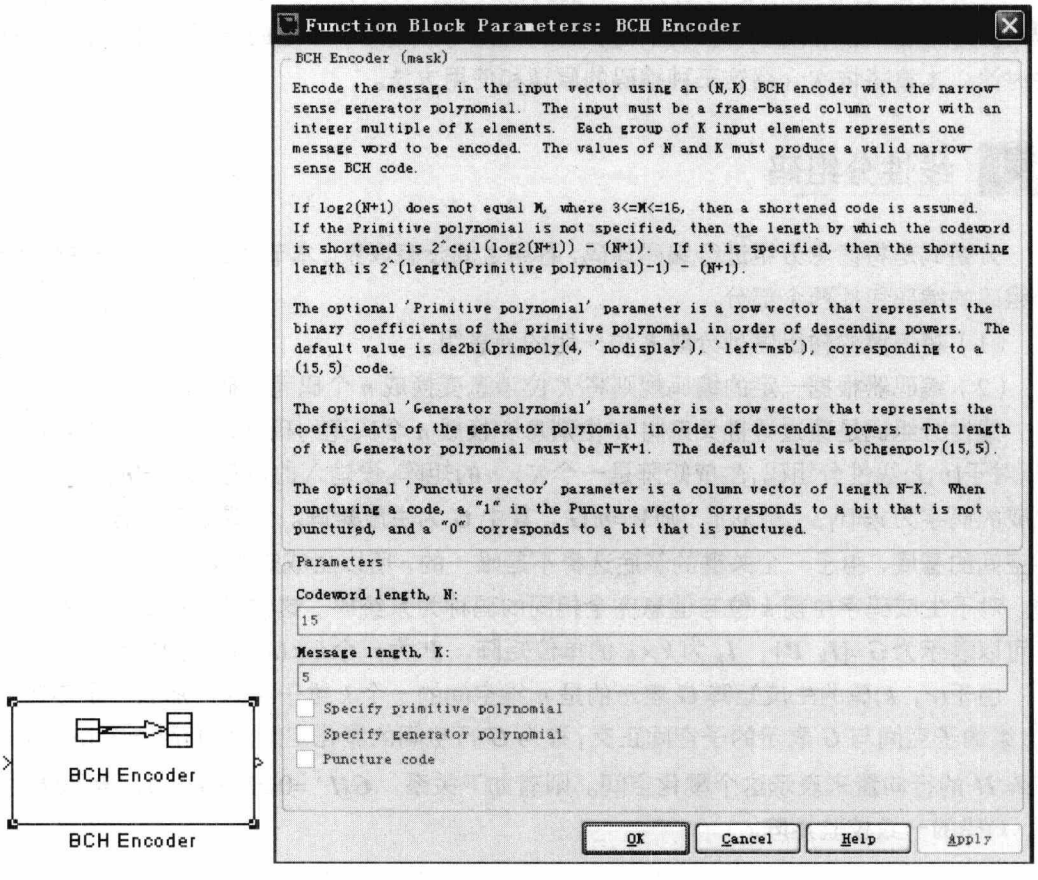


图 8-1　BCH 编码模块及其参数设定框

BCH 编码模块中包含若干参数项，下面分别简单介绍：

Codeword length, N：设定码字长度，同时也是输出向量的长度。

Message length, K：设定信息长度，同时也是输入向量的长度。

Specify primitive polynomial：选定后显示“Primitive polynomial”项。

Primitive polynomial：代表初始多项式的二进制系数的行向量。默认值为 de2bi(primpoly(4,'nodisplay'),'left-msb')，符合(15,5)码。本项只有当“Specify primitive polynomial”项选定后才显示。

Specify generator polynomial：选定后显示“Generator polynomial”项。

Generator polynomial：代表初始多项式的二进制系数的行向量。默认值为 bchgenpoly(15,5)。长度为 N-K+1。只有选定“Specify generator polynomial”后才有效。

Puncture code：选定后显示“Puncture vector”项。

Puncture vector：长度为 N-K 的列向量。1 表示未打孔位，0 表示打孔位。默认值为 [ones(8,1)；zeros(2,1)]。

8.1.1.2　BCH 译码模块

BCH 译码模块用于对输入的经过 BCH 编码的信息进行译码。BCH 译码模块的输入必须是有 N（the number of punctures）的整数倍个元素的基于帧的列向量。每 N 个输入元素代表一个译码信号字。模块中完成(N,K)的 BCH 译码。其中 N 具有（2^M-1）的形式，$3 \leqslant M \leqslant 16$。如果 N 小于（2^M-1），那么模块认为码长减小了 2^M-1-N；如果 N 大于（2^M-1），那么必须在模块参数项“Primitive polynomial”中设定适当的 M 值。模块中 K 的有效值与 N 对应，最大为 511。

BCH 译码模块及其参数设定框如图 8-2 所示。

BCH 译码模块中包含若干参数项，下面分别简单介绍：

Codeword length, N：设定码字长度，同时也是输入向量的长度。

Message length, K：设定信息长度，同时也是第一个输出向量的长度。

Specify primitive polynomial：选定后显示“Primitive polynomial”项。

Primitive polynomial：代表初始多项式的二进制系数的行向量。默认值为 de2bi(primpoly(4,'nodisplay'),'left-msb')，符合(15,5)码。只有选定“Specify primitive polynomial”后才有效。

Specify generator polynomial：选定后显示“Generator polynomial”项。

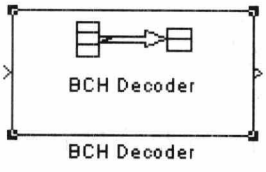
Generator polynomial：代表初始多项式的二进制系数的行向量。默认值为 bchgenpoly(15,5)。长度为 N-K+1。本项只有当“Specify generator polynomial”项选定后才显示。

Puncture code：选定后显示“Puncture vector”项。

Puncture vector：长度为 N-K 的列向量。1 表示未打孔位，0 表示打孔位。默认值为 [ones(8,1)；zeros(2,1)]。

Enable erasures input port：选定后打开 Era 和 Err 两个端口，通过 Era 端口，可以输入与输入码字长度相同的基于帧的列向量。Err 端口用于输出修正的错误数。

Output number of corrected errors：选定后模块会另外提供一个输出端口，用于输出模块探测到的输入码字的错误数。



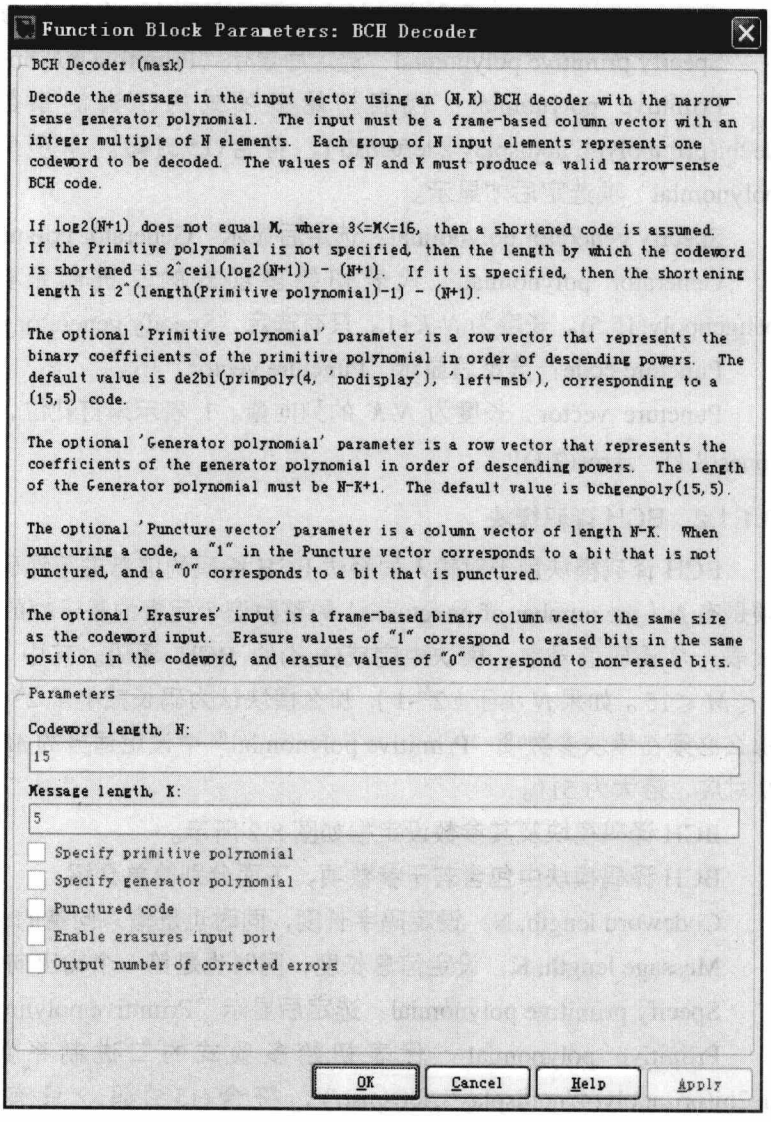


图 8-2　BCH 译码模块及其参数设定框

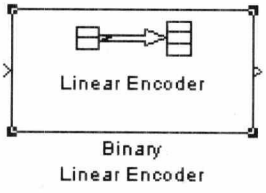
8.1.2　二进制线性编码/译码

8.1.2.1　二进制线性编码模块

二进制线性编码模块使用生成矩阵进行二进制线性编码，如果 K 为信息长度，那么 Generator matrix 应该有 K 行，如果 N 为码字长度，那么 Generator matrix 应该有 N 列。

二进制线性编码模块的输入必须有 K 个元素，如果输入是基于帧的，那么它必须是一个列向量，输出必须是长度为 N 的向量。

二进制线性编码模块及其参数设定框如图 8-3 所示。



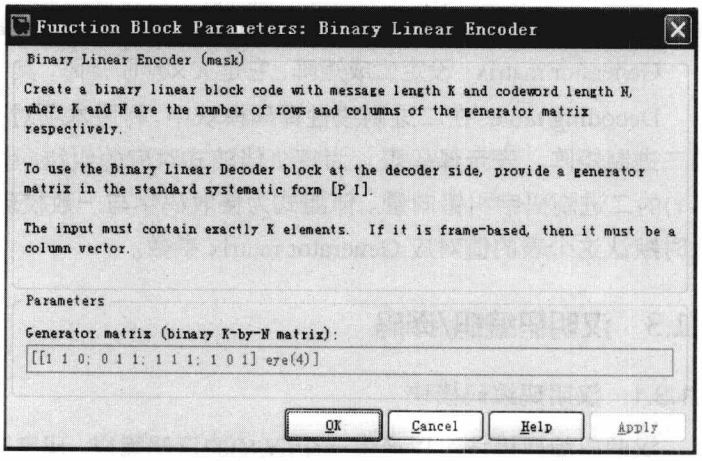


图 8-3　二进制线性编码模块及其参数设定框

二进制线性模块中包含一个参数项：

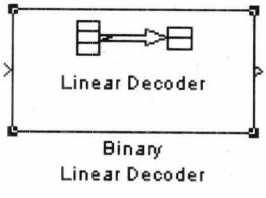
Generator matrix：设定生成矩阵，它是 $K \times N$ 阶矩阵，其中 K 为信息长度，N 为码字长度。

8.1.2.2　二进制线性译码模块

二进制线性译码模块完成对二进制码字向量的线性译码。参数 Generator matrix 为模块的生成矩阵，它应该与二进制线性编码模块中的 Generator matrix 参数相同。如果 K 为信息长度，那么 Generator matrix 应该有 K 行，如果 N 为码字长度，那么 Generator matrix 应该有 N 列。

模块的输入必须有 N 个元素，如果输入是基于帧的，那么它必须是一个列向量，输出必须是长度为 K 的向量。

二进制线性译码模块及其参数设定框如图 8-4 所示。



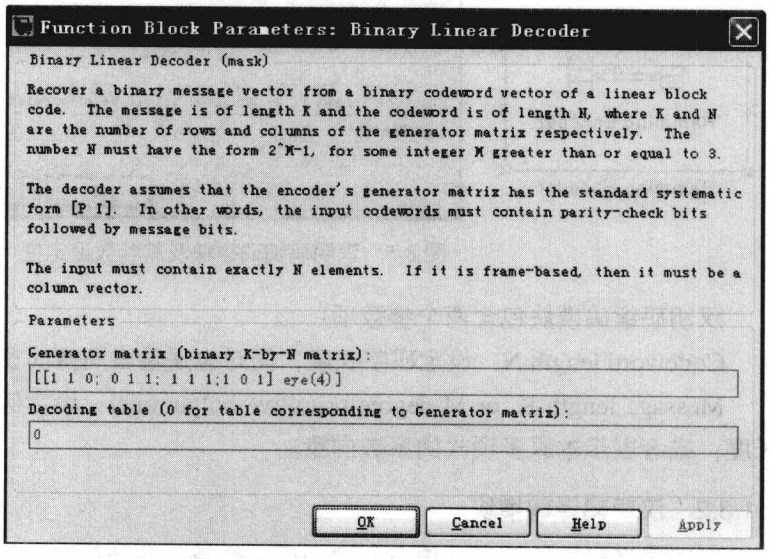


图 8-4　二进制线性译码模块及其参数设定框

二进制线性译码模块中包含两个参数项，下面分别做简单介绍：

Generator matrix：设定生成矩阵，它是 $K \times N$ 阶矩阵，其中 K 为信息长度，N 为码字长度。

Decoding table：在二进制线性译码模块中，利用本项对码字进行纠错。可以为 $2^{N-K} \times N$ 的二进制矩阵，表示译码表，为每个伴随式对应的纠错向量。矩阵的第 r 行是伴随式等于 $(r\text{-}1)$的二进制码字纠错向量。伴随式为接收码字与一致校验矩阵转置的乘积。可以设为 0，此时默认这个表的值对应 Generator matrix 参数。

8.1.3 汉明码编码/译码

8.1.3.1 汉明码编码模块

汉明码编码模块可以产生一个(N,K)的汉明编码。码字长度 $N=2^M-1$，其中 M 为大于等于 3 的整数。信息长度 $K=M-N$。

在选定编码方案时，可以指定本原多项式或使用默认设置。使用默认设置时，输入 N 和 K 作为输入参数，模块将使用 gfprimdf(M)作为 GF(2^M)的本原多项式。如果使用指定本原多项式的方式，那么第一项输入 N 作为参数，第二项输入一个二进制向量作为参数指定本原多项式。这个向量以升幂的顺序列出本原多项式的系数。另外可以通过通信工具箱中的 gfprimdf 函数来求本原多项式。

汉明码编码模块及其参数设定框如图 8-5 所示。

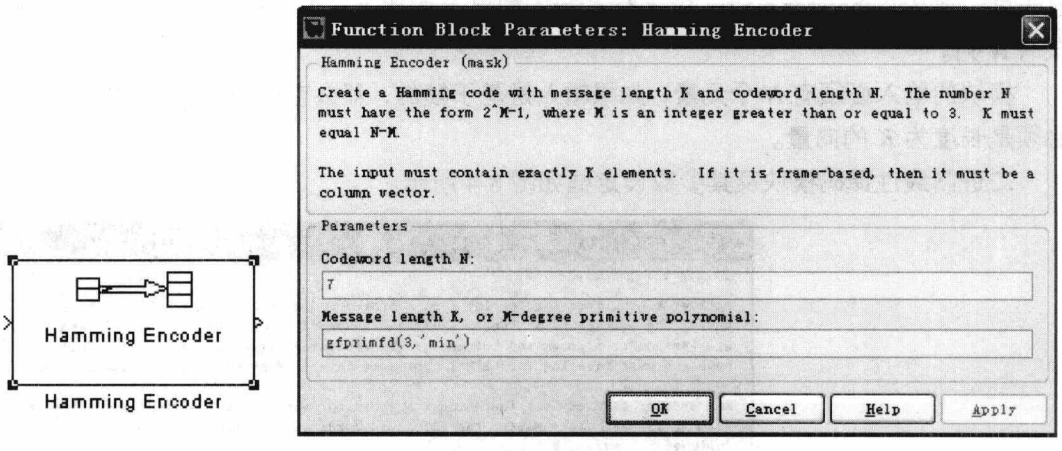


图 8-5 汉明码编码模块及其参数设定框

汉明码编码模块包含两个参数项：

Codeword length N：设定码字长度，同时也是输出向量的长度。

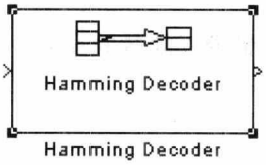
Message length K, or M-degree primitive polynomial：设定信息长度，也是输入向量的长度；或者表示本原多项式的系数向量。

8.1.3.2 汉明码译码模块

汉明码译码模块用于从接收的汉明码中恢复出原始信息。为了能够正确译码，模块所有的参数必须与相应的汉明码编码模块的参数相匹配。

在选定编码方案时，可以指定本原多项式或使用默认设置。使用默认设置时，输入 N 和 K 作为输入参数，模块将使用 gfprimdf(M)作为 GF(2^M)的本原多项式。如果使用指定本原多项式的方式，那么第一项输入 N 作为参数，第二项输入一个二进制向量作为参数指定本原多项式。这个向量以升幂的顺序列出本原多项式的系数。另外可以通过通信工具箱中的 gfprimdf 函数来求本原多项式。

汉明码译模块及其参数设定框如图 8-6 所示。



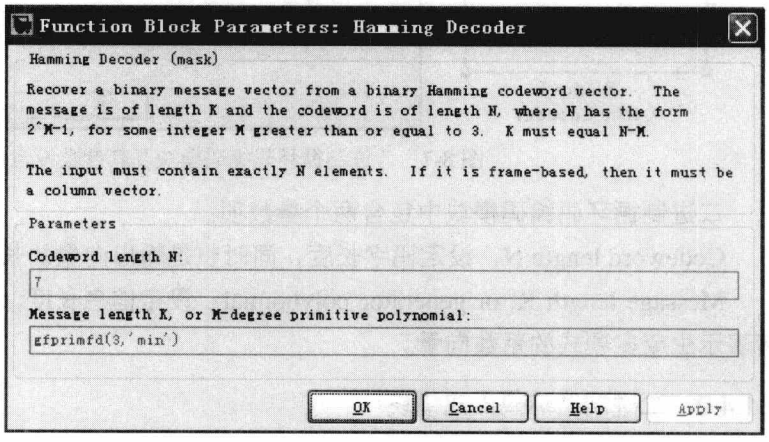


图 8-6　汉明码译模块及其参数设定框

汉明码译模块包含两个参数项：

Codeword length N：设定码字长度，同时也是输出向量的长度。

Message length K, or M-degree primitive polynomial：设定信息长度，也是输入向量的长度；或者表示本原多项式的系数向量。

8.1.4　二进制循环码编码/译码

8.1.4.1　二进制循环码编码模块

二进制循环码编码模块用于对输入的二进制信息进行循环编码。循环码可由它的生成多项式唯一确定。(N,K)循环码的码长 $N=2^M-1$，其中 M 为大于等于 3 的整数。信息位为 K，$K<N$。模块的输入数据长度为 K，输出数据长度为 N。如果输入为基于帧的，则必须为列向量结构。

循环码的编码方案有下面两种方式：

（1）为了生成一个(N,K)循环码，将 N 和 K 作为第一个和第二个输出参数。模块会计算一个合适的生成多项式 cyclpoly(N,K, 'min')。

（2）为了编一个码长为 N，并且具有特定（N-K）阶二进制生成多项式的循环码，输入 N 作为第一个参数，输入一个二进制向量作为第二个参数。这个向量表示生成的多项式，它是以升幂的顺序排列的参数。另外可以通过通信工具箱中的 cyclpoly 函数来产生循环码的生成多项式。

二进制循环码编码模块及其参数设定框如图 8-7 所示。

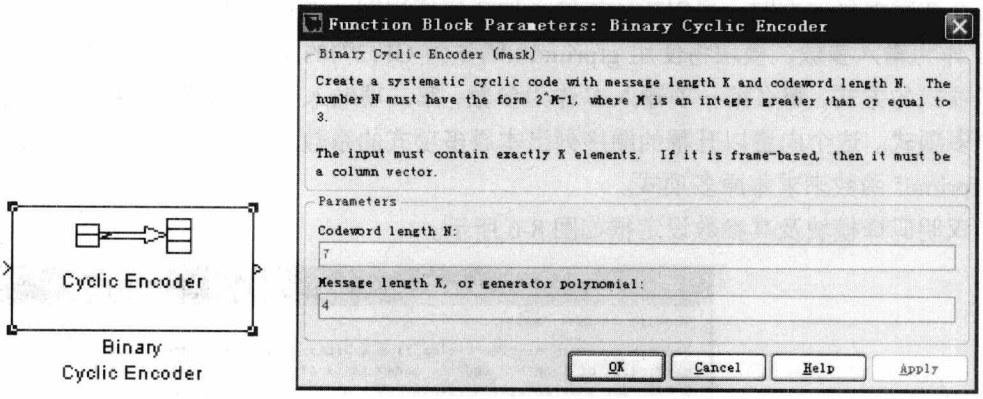


图 8-7　二进制循环码编码模块及其参数设定框

二进制循环码编码模块中包含两个参数项：

Codeword length N：设定码字长度，同时也是输出向量的长度。

Message length K, or generator polynomial：设定信息长度，也是输入向量的长度；或者表示生成多项式的系数向量。

8.1.4.2　二进制循环码译码模块

二进制循环码译码模块用于对输入的循环码进行译码。(N,K)循环码的码长 $N=2^M-1$，其中 M 为大于等于 3 的整数。输入必须包含 N 个元素。如果输入为基于帧的，则必须为列向量结构。

循环码的译码方案有下面两种方式：

（1）为了译码一个(N,K)循环码，将 N 和 K 作为第一个和第二个输出参数。模块会计算一个合适的生成多项式 cyclpoly(N,K, 'min')。

（2）为了译码一个码长为 N，并且具有特定（N-K）阶二进制生成多项式的循环码，输入 N 作为第一个参数，输入一个二进制向量作为第二个参数。这个向量表示生成的多项式，它是以升幂的顺序排列的参数。另外可以通过通信工具箱中的 cyclpoly 函数来产生循环码的生成多项式。

二进制循环码译码模块及其参数设定框如图 8-8 所示。

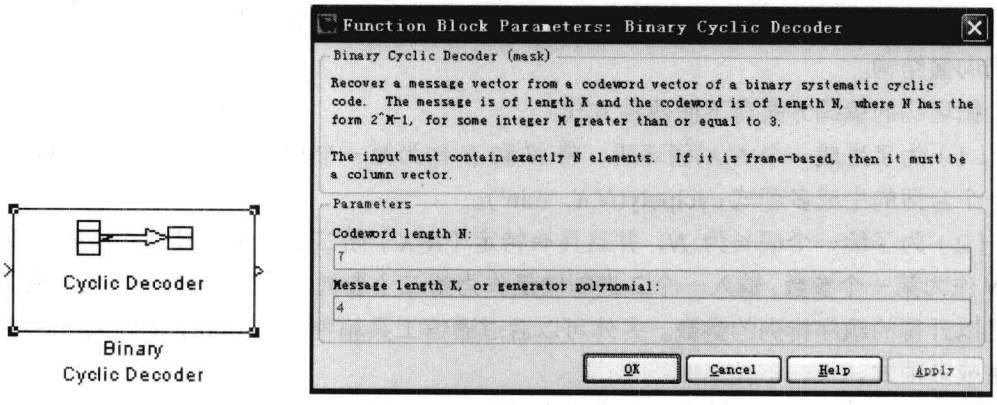


图 8-8　二进制循环码译码模块及其参数设定框

二进制循环码译码模块中包含两个参数项：

Codeword length N：设定码字长度，同时也是输入向量的长度。

Message length K, or generator polynomial：设定信息长度，也是输出向量的长度；或者表示生成多项式的系数向量。

8.2 循环卷积码

8.2.1 卷积码编码器原理

卷积码是纠错编码的一种特殊情况，它与分组码不同，卷积码的编码器不是无记忆的设备。对卷积码编码器输入相同的符号时，输出可能不同，因为卷积码的编码不仅与当前输入有关，而且与以前的输入有关。下面具体介绍一下卷积码编码器的两种描述方式。

8.2.1.1 卷积码编码器的多项式描述

卷积码编码器的多项式描述，描述了卷积码编码器中移位寄存器与模二加法器的连接关系。图 8-9 中描述了一个单个输入、两个输出和两个移位寄存器的前反馈卷积码编码器。

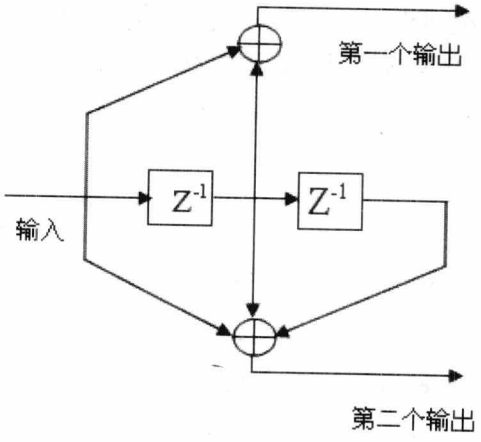


图 8-9　前反馈卷积码编码器框图

卷积码编码器的多项式描述包含两个或三个部分（取决于是前反馈编码器还是反馈编码器）。三个部分分别为：约束长度、生成多项式、反馈连接多项式。下面对这三部分做简单介绍。

（1）约束长度

编码器的约束长度形成一个向量，这个向量的长度为编码器的输入个数，向量的元素表示存储在每个移位寄存器中的比特数，包括当前输入比特。图 8-9 中的约束长度为 3。是一个标量，因为编码器只有一个输入。它的值等于 1 加上此输入的移位寄存器的个数。

（2）生成多项式

如果编码器有 k 个输入，n 个输出，那么这个码的生成矩阵为一个 $k \times n$ 的矩阵。在第

i 行第 j 列的元素，表示第 i 个输入如何影响第 j 个输出。对于一个系统反馈编码器的系统位，它生成矩阵的项与反馈连接矢量的相应元素匹配。

在其他情况下，可以按照下面的方法来决定生成矩阵的(i,j)项。

① 二进制表示。如果一个移位寄存器接到加法器上，则在相应位用 1 表示，没有连接用 0 来表示。在二进制数中，最左边的数表示当前输入，最右边的数表示移位寄存器中保存最久的输入。

② 八进制表示。将二进制的数转换成八进制的数，从最右边的数开始，最右边的位为最低位，如果二进制的数不是 3 的倍数，那么按照需要在左边补零。

（3）反馈连接多项式

如果描述一个带反馈的编码器，用户需要一个反馈连接向量。这个向量的长度为编码器的输入个数。向量的元素用八进制数的形式表示每个输入的反馈连接。

如果编码器具有反馈结构，而且是系统的，那么生成多项式和反馈连接的多项式相应的系统参数一定是相同值。

8.2.1.2 卷积码的网格描述

卷积码的网格描述表示出编码器每种可能的输入如何影响输出和编码器的状态改变。

卷积码编码器的网格图如图 8-10 所示。这个编码器有四个状态，一位的输入，两位的输出，是一个编码效率为 1/2 的编码器。在图中每条实线表示当前输入为 0 时，编码器如何改变状态；每条虚线表示当前输入为 1 时，编码器如何改变状态。每条线上的八进制数表示当前编码器的输出。

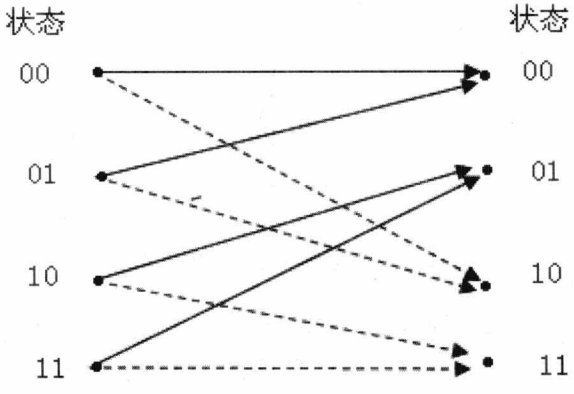


图 8-10　四个状态的卷积编码器网格图

注意：任何卷积码编码器的多项式描述等同于一个网格描述，但是一些网格描述并没有相对应的多项式描述。

在 MATLAB 中表示网格，使用的是一种叫做网格结构的数据，这种网格结构必须包含 5 个域，如表 8-1 所示。

表 8-1　编码效率 *k*/*n* 的网格结构的域

网格结构的域	维　数	含　义
NumInputSymbols	标量	输入符号数为 2^k
NumOutputSymbols	标量	输入符号数为 2^n
NumStates	标量	状态数
NextStates	NumStates × 2^k 的矩阵	当前状态，当前输入下，所有可能的下一个状态
Outputs	NumStates × 2^k 的矩阵	当前状态，当前输入下，所有可能的下一个输出（十进制）

在矩阵 NextStates 中，每一项是一个 0～（NumStates-1）的整数，它的第 i 行第 j 列表示开始状态为 i-1 时的下一个状态。在将输入位转换为十进制形式时，将第一个输入位作为最高位。

在输出矩阵中，第 i 行第 j 列的元素表示开始状态为 i-1 且输入位十进制形式为 j-1 时的编码器输出。在将输入位转换为十进制形式时，与 NextStates 矩阵的情况相同。

当已经知道要放入每个域的信息时，用户可以用下列方式建立一个网格结构：

（1）分别定义 5 个域，使用 structurename.fieldname 的形式。

（2）把所有的域名字和值收集到一个单独的结构命令中。

（3）先用多项式描述方式来描述编码器，然后使用 poly2trellis 函数将其转换成一个有效的网格结构。

通信模块库中提供了两种卷积码的译码器：后验概率解码器和 Viterbi 解码器。下面对它们做简单介绍。

8.2.2　后验概率解码器

8.2.2.1　功能与原理

后验概率解码器模块用于完成卷积码后的验概率译码。该模块包含两个输入端：输入端 $L(u)$表示与解码对应的编码器的输入信息比特的对数概率；输入端 $L(c)$表示码字比率的对数概率。同时该模块有两个输出端：L(u)和 L(c)，分别表示对输入端 L(u)和 L(c)的纠正。

如果卷积码使用有 2^n 个可能值的符号，那么模块的 L(c)向量长度为 $Q \times n$。同样的，如果被译码的数据有 2^K 个可能的符号，那么模块的 L(u)向量长度为 $Q \times k$。其中 Q 为每个时间步模块处理的帧数。当输入是基于采样的信号时，Q =1。

如果只需要输入端 L(c)和输出端 L(u)，那么可以将输入端 L(u)接 Ground 模块，将输出端 L(c)接 Terminator 模块。

8.2.2.2　参数项说明

后验概率解码器模块及其参数设定框如图 8-11 所示。

后验概率解码器模块包含若干参数，下面简单介绍一下：

Trellis structure：卷积码的网格描述。要与编码器的该项参数一致。

Termination method：一个复选框，可选 Truncated 和 Terminated。表示卷积码编码器在帧开始和结束时如何表示处理网格。

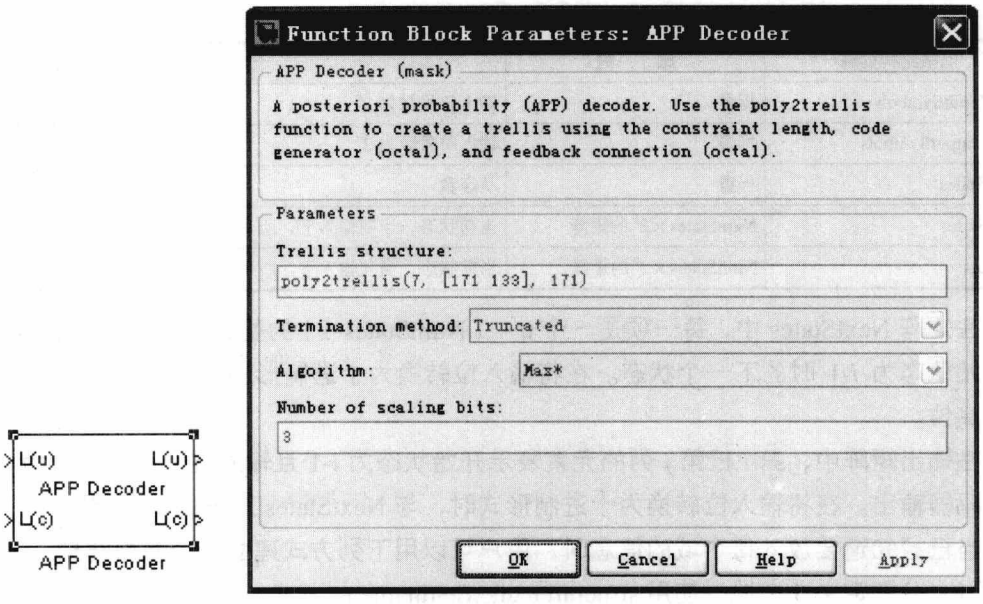


图 8-11　后验概率解码器模块及其参数设定框

Algorithm：一个复选框，可选 True APP、Max*或 Max。用户可以通过本参数来控制部分译码算法。选项 True APP 实现后检验概率。为了提高速度，选项 Max*和 Max 可以近似表示为 $\log\sum_i e^{a_i}$。Max 选项使用 $\max\{a_i\}$作为近似值，而 Max*选项使用 $\max\{a_i\}$外加一个修正项 $\ln(1+\exp(-|a_{i-1}-a_i|))$。

Number of scaling bits：为一个 0～8 的整数，表示译码器为了避免精度损失使用多少个比特衡量数据。只有当“Algorithm”选择 Max*时本项才有效。

8.2.3　Viterbi 解码器

8.2.3.1　功能与原理

Viterbi 解码器模块对输入信号进行译码，产生二进制输出信号。该模块可在同一时刻处理多个模块。

如果编码器收到 n 个比特流的输入，那么模块的输入向量长度为 $L*n$；如果编码器输出 k 个比特流，那么模块的输出向量长度为 $L*k$。当输入基于采样时，L=1；当输入基于帧时，L 可以是任意的正整数。

输入数据的类型可以由模块中的“Decision type”决定，表 8-2 给出了该参数项的取值和含义。

表 8-2　Decision type 项取值表

参数 Decision type 值	译码器可能的输入元素	值的含义
Unquantized	实数	+1：逻辑 0 -1：逻辑 1

续表

参数 Decision type 值	译码器可能的输入元素	值的含义
Hard Decision	0 和 1	0：逻辑 0 1：逻辑 1
Soft Decision	0～(2^b–1)之间的整数， b 为参数“Number of soft decision bits”的值	0：最可能判断为逻辑 0 2^b–1：最可能判断为逻辑 1 中间的值越接近 0 表示越可能判断为逻辑 0；越接近 2^b–1 表示越可能判断为逻辑 1

如果输入的信号是基于帧的，那么模块有三种可能的方式在连续的帧之间转换。参数“Operation mode”项决定具体模式的选取。三种可能的方式为：

（1）Continuous 模式，模块在每一帧的末端存储中间状态距离，以便下一帧的使用。每个回溯路径被单独处理。

（2）Truncated 模式，模块单独处理每一帧。回溯路径从最佳距离状态开始，一直到全零状态结束。这种模式在“Operation mode”项设定为 Truncated (reset every frame)时有效。

（3）Terminated 模式，模块单独处理每一帧。回溯路径从全零状态开始，也结束于全零状态。使用这个模式时，未编码信号末端必须具有足够的 0。如果信息组长度为 k，编码器的长度为 constr，那么未编码信号末端至少有 k*max(constr-1)个 0 才可以使用这个模式。

在特殊情况下，当基于帧的输入信号只包含一个字符时，最适合 Continuous 模式。

“Traceback depth”参数项 D 影响译码的延时。译码的延时是指输入第一个符号前的 0 符号数，如果输入信号是基于采样的，那么译码延时包含 D 个 0 符号。如果输入信号基于帧，而且参数“Operation mode”项设定为 Continuous 模式时，译码延时包含 D 个 0 符号。如果“Operation mode”项设定为 Truncated 或 Terminated 模式时，输出没有时延，并且参数“Traceback depth”必须小于等于每个符号数。

对于编码效率为 1/2 的卷积码，典型的“Traceback depth”值为编码约束长度的 5 倍。只有当参数“Operation mode”项设定为 Continuous 模式时，重置端才可用。选定“Enable reset input port”将会增加一个标有 Rst 的输入端口，当 Rst 输入不为 0 时，译码器通过下面三种操作返回初始状态：将全零状态的距离设为 0、将其他状态的距离设为最大值、将回溯存储设为 0。

8.2.3.2　参数项设定

Viterbi 解码器模块及其参数设定框如图 8-12 所示。

如图 8-12 所示，Viterbi 解码器参数设定框中包含“Main”和“Data Types”两类，默认为“Main”类。下面分别对各类中的参数项做简单介绍。

1.“Main”类参数项说明

Trellis structure：卷积码的网格描述。该项要与编码器中对应项参数一致。

Punctured code：选定后显示“Punctured code”项。

Puncture vector：发射器或编码器中的常打孔模式矢量。格式为 0s 和 1s，其中 0s 代表打孔位。

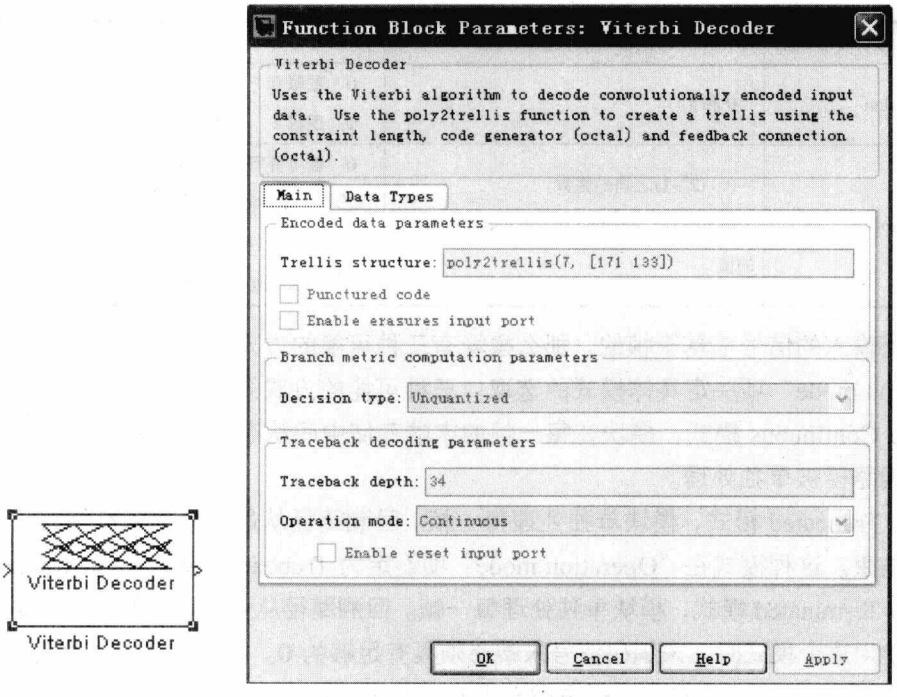


图 8-12　Viterbi 解码器模块及其参数设定框

Enable erasures input port：选定该项后，增加一个标有 Rst 的输入端口。通过此端口，可以输入具有 0s 和 1s 格式的删除矢量。1s 代表删除位。对于输入数据流中的这些删除位，译码器将不会更新它的分支量度。

Decision type：为一复选框，包含 Unquantized、Hard Decision 和 Soft Decision 三项。

Number of soft decision bits：软判决中用来表示输入的比特数。只有当“Decision type”项设定为 Soft Decision 时本项有效。

Error if quantized input values are out of range：量化的输入值超出范围后的出错项。只有当“Decision type”项设定为 Soft Decision 或 Hard Decision 时本项有效。

Traceback depth：设置用来构建每个回溯路径的网格分支数。

Operation mode：表示在连续输入帧之间的转换方式。有 Continuous、Terminated 和 Truncated 三种模式。对于基于采样的输入，必须采用 Continuous 模式。

Enable reset input port：选中本项后，译码器模块会出现标有 Rst 的输入端口，当这个端口的输入为非零值时，模块对其状态进行复位，将其中间记忆恢复到初始状态。

2.“Data Types”类参数项说明

选定“Data Types”类后，参数设定框如图 8-13 所示。

本类中只包含“Output data type”参数项。用来设定输出信号的数据类型。可以是 double、single、boolean、int8、uint8、int16、uint16、int32、uint32 或设置为 Inherit via internal rule 或 Smallest unsigned integer。

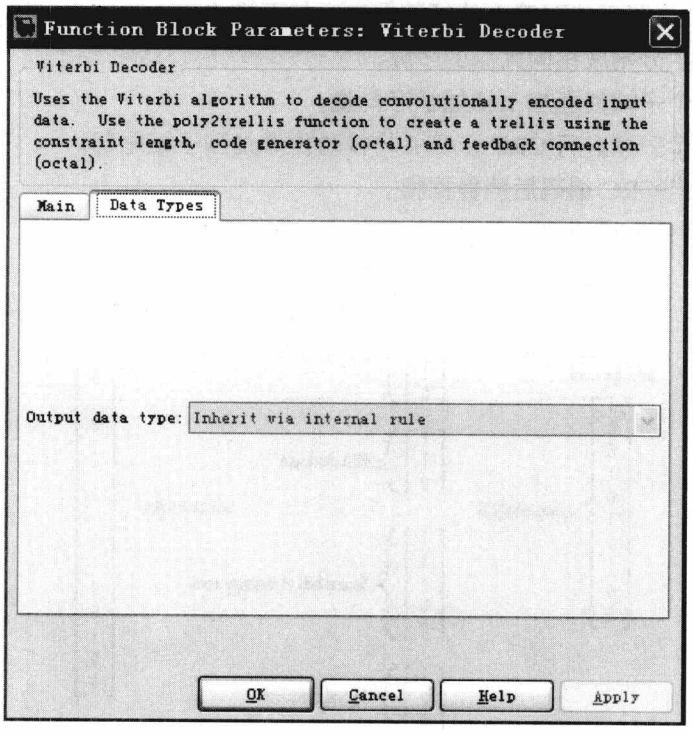


图 8-13　“Data Types”类参数设定框

8.3　CRC 循环冗余码校验

循环冗余码是一种使用相当频繁的校验码。与分组码和卷积码不同，循环冗系码不具有纠错能力。当接收端检测到传输错误时，它不是不去纠正这个错误，而是要求发送端重新发送这个信号序列。在循环冗余码的编码过程中，发送端对每一个特定长度的信息序列计算得到一个循环冗余码，并把这个循环冗余码附加到原始的信息序列的结尾一起发送出去。接收端接收到带有循环冗余码的信号后，从中分离出信息位序列和循环冗余码，然后根据接收到的信息位序列重新计算循环冗余码。如果这个重新计算得到的循环冗余码与分离出来的循环冗余码不同，则接收信号序列存在着传输错误。此时接收端会要求发送端重新发送这个信号序列，通过这个过程实现对信号的纠错。

在 MATLAB 中，CRC 产生器有两种，即常规 CRC 产生器和 CRC-N 产生器，这两个 CRC 产生器比较接近，它们之间的区别在于，后者提供了 6 个常用的 CRC 生成多项式，使用起来比较方便。本节对这两个产生器及校验进行简单介绍。

8.3.1　常规 CRC 产生器

常规 CRC 产生器根据输入的一帧数据计算得到这帧数据的循环冗余码 CRC，并且把这个循环冗余码附加到帧数据的后面，形成输出数据流。其输入输出格式必须是二进制列向量。

常规 CRC 产生器模块对输入的帧数据进行如下操作：

（1）将输入帧分割成相同大小的子帧。

（2）在每一个子帧前面加上初始状态向量。

（3）对每一个子帧进行 CRC 运算并将校验位添加到子帧尾部。

（4）最后输出 CRC 编码后的数据帧。

常规 CRC 产生器模块的执行过程可用图 8-14 表示。

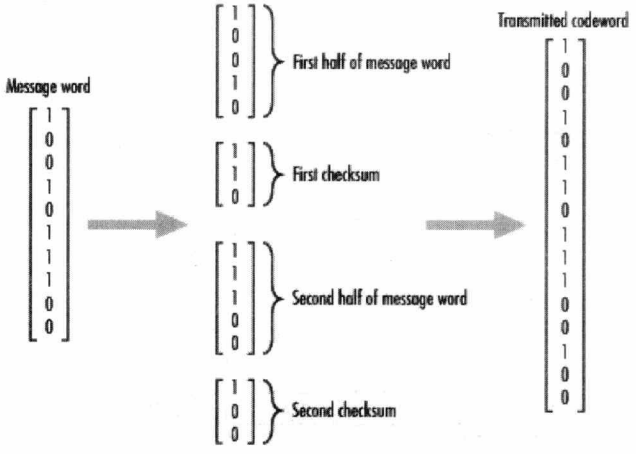


图 8-14　常规 CRC 产生器模块的工作流程

图 8-14 中的输入帧长度为 10，生成多项式的阶数为 3，初始状态为 0，每一个子帧被分成两个子帧。模块首先将输入帧数分割成长度为 5 的子帧，在每个子帧后面添加 3 位 CRC 检验位，然后将两个子帧进行串接，连续输出，输出帧长为 5+3+5+3=16。

常规 CRC 产生器模块及其参数设定框如图 8-15 所示。

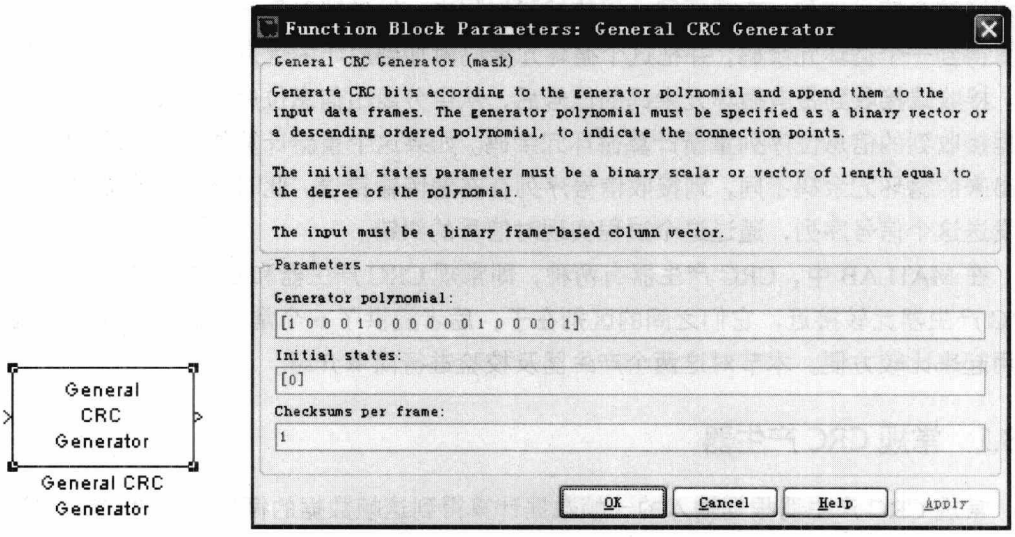


图 8-15　常规 CRC 产生器模块及其参数设定框

常规 CRC 产生器模块中包含若干参数项，下面分别做简单介绍：

Generator polynomial：指定 CRC 运算的生成多项式，是按照递减顺序排列的向量。如果是二进制的，那么每个数值代表生成多项式中幂的系数，比如[1 1 0 1]表示生成多项式为 x^3+x^2+1。若为整数，代表非 0 次幂，如生成多项式 x^3+x^2+1 也可以表示为[3 2 0]。

Initial states：用于确定常规 CRC 产生器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于常规 CRC 产生器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于常规 CRC 产生器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

Checksums per frame：指定每帧数据产生的校验和的个数。如果每帧的校验和的个数等于 k，那么每帧输入数据的长度应该是 k 的整数倍。

8.3.2　CRC-N 信号产生器

CRC-N 信号产生器计算每一个输入帧的循环冗余码，并且把计算得到的循环冗余码附加到输入帧的末尾。模块中的输入输出必须是帧结构的二进制向量。

CRC-N 信号产生器模块及其参数设定框如图 8-16 所示。

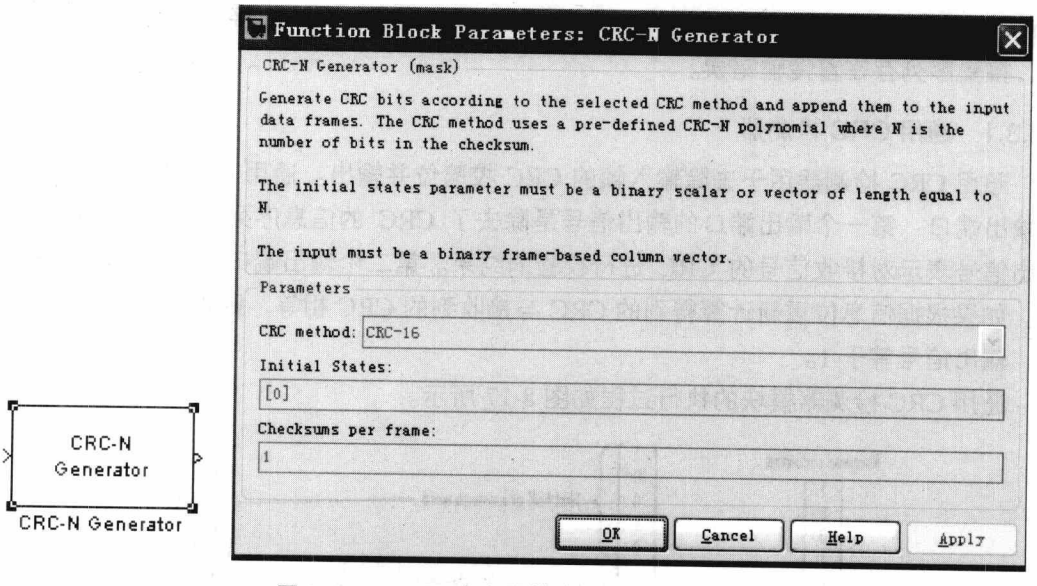


图 8-16　CRC-N 产生器模块及其参数设定框

CRC-N 产生器模块中包含若干参数项，下面分别做简单介绍：

CRC-N method：指定 CRC 校验的生成多项式。共有 6 个选项，不同的选项对应的生成多项式和校验位长度如表 8-3 所示。

Initial states：用于确定 CRC-N 产生器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于 CRC-N 产生器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于 CRC-N 产生器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

表 8-3　CRC 生成多项式与校验位长度关系表

CRC 模式	生成多项式	校验位长度
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	32
CRC-24	$x^{24}+x^{23}+x^{14}+x^{12}+x^6+1$	24
CRC-16	$x^{16}+x^{15}+x^2+1$	16
Reversed CRC-16	$x^{16}+x^{14}+x+1$	16
CRC-8	$x^8+x^7+x^6+x^4+x^2+1$	8
CRC-4	$x^4+x^3+x^2+x+1$	4

Checksums per frame：指定每帧数据产生的校验和的个数。如果每帧的校验和的个数等于 k，那么每帧输入数据的长度应该是 k 的整数倍。

8.3.3　CRC 冗余码校验

产生器模块生成冗余码后，还要对其进行校验。对应于两种生成器 MATLAB 还提供了两种检测器：通用 CRC 检测器和 CRC-N 检测器。这两种检测器具有相同的工作原理，它们首先从接收到的二进制序列中分离出信息序列和 CRC，然后根据接收端的信息序列重新计算 CRC。如果重新计算的结果与接收到的 CRC 相等，则认为接收序列是正确的；否则，接受序列存在着传输错误。

8.3.3.1　通用 CRC 检测器

通用 CRC 检测器用于去除输入帧的 CRC 校验位并输出。通用 CRC 检测器模块有两个输出端口：第一个输出端口的输出信号是除去了 CRC 的信息序列；第二个输出端口的输出信号表示对接收信号的 CRC 进行校验的结果。第二个输出端口的输出信号是一个向量。如果根据信息位重新计算得到的 CRC 与接收到的 CRC 相等，则输出信号等于 0；否则，输出信号等于 1。

通用 CRC 检测器模块的执行过程如图 8-17 所示。

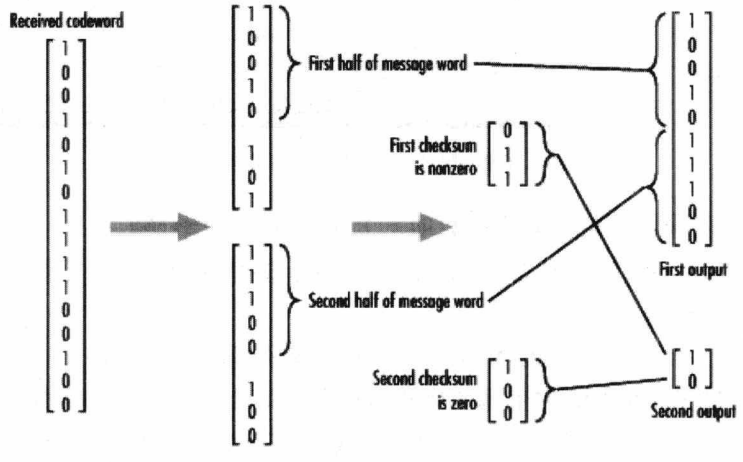


图 8-17　通用 CRC 检测器模块的执行过程图

图 8-16 中的接收信息码长为 16，生成多项式的次数为 3，内部寄存器的初始状态为 0，子帧个数为 2。模块首先将输入的数据分成长度相同的两个子帧，再去除每个子帧后面的 3bitCRC 校验信息。图 8-16 中并未显示初始状态，因为 0 状态并不会影响到模块的 CRC 运算。模块的第一个输出端输出去除校验位后的信息流，两个子帧相加正好等于 10bit。第二个输出端输出为[1 0]，表示第一个子帧的校验位与已知校验信息不同，也就是在传输过程中出现了错误。第二个子帧校验位与已知的校验信息相同，表示传输中没有出现错误。

通用 CRC 检测器模块及其参数设定框如图 8-18 所示。

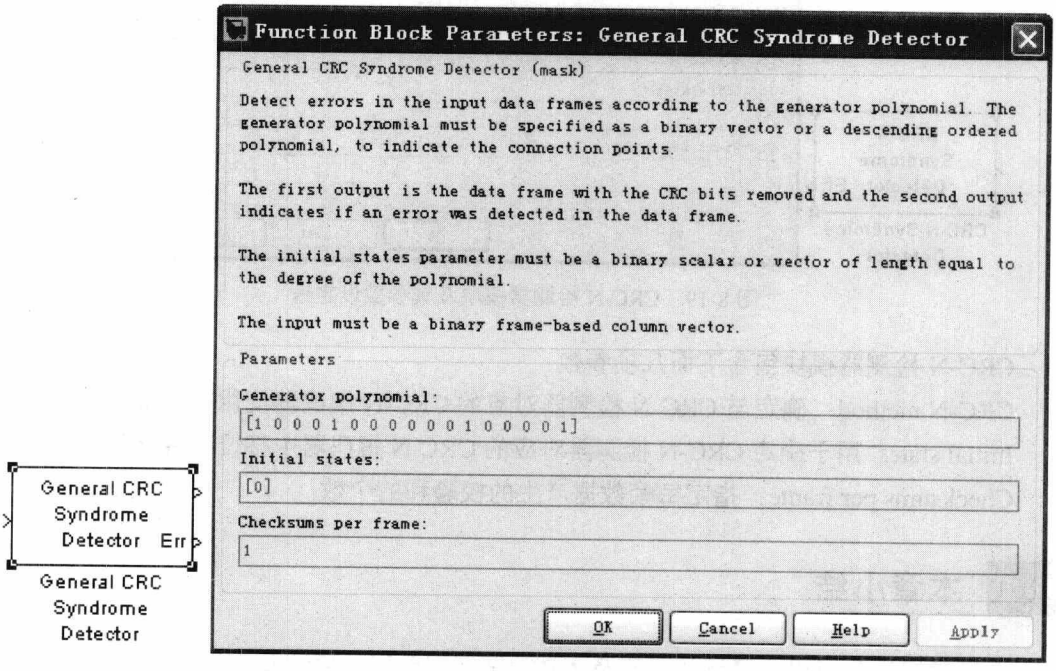


图 8-18　通用 CRC 检测器模块及其参数设定框

Generator polynomial：与通用 CRC 检验器对应的 CRC 编码器的生成多项式。可以是二进制向量或整型向量的形式。

Initial states：用于确定通用 CRC 检测器对应的 CRC 编码器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于通用 CRC 产生器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于通用 CRC 产生器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

Checksums per frame：指定每帧数据产生的校验和的个数。

8.3.3.2　CRC-N 检测器

CRC-N 检测器从接收到的信号中分离出信息序列和 CRC，然后对信息序列重新计算 CRC。如果重新计算得到的 CRC 与分离出来的 CRC 相等，那么认为接收到的信号是正确的，否则接收到的信号存在着传输错误。

CRC-N 检测器模块及其参数设定框如图 8-19 所示。

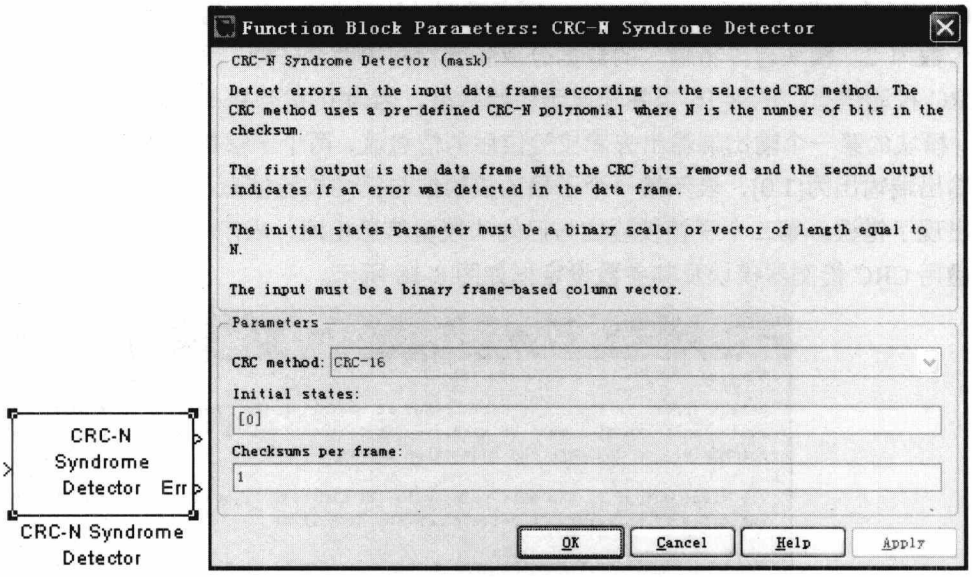


图 8-19　CRC-N 检测器模块及其参数设定框

CRC-N 检测器模块包含下面几项参数：

CRC-N method：确定与 CRC-N 检测器对应的 CRC-N 编码器使用的生成多项式。

Initial states：用于确定 CRC-N 检测器对应的 CRC-N 编码器中移位寄存器的初始状态。

Checksums per frame：指定每帧数据产生的校验和的个数。

8.4 本章小结

本章对线性分组码、循环卷积码及 CRC 循环冗余码的原理及应用做了细致介绍，并对 MATLAB 中对应的编码译码器做了详细的讲解。读者学习的时候，需要重点理解各码的编译原理，比较它们的不同点，同时熟悉掌握各模块的参数含义。

第 9 章　同步

在通信系统中，同步具有非常重要的作用。所谓同步就是收发双方在时间上步调一致，在频率和相位上也一致。同步是信息传递的前提，通信系统能否有效可靠的工作，在很大程度上依赖于有无良好的同步系统。

本章中主要介绍 MATLAB 中和同步相关的一些仿真模块，包括载波相位恢复、定时恢复、基本锁相环及压扩振荡器模块等。

9.1 载波相位恢复

9.1.1 CPM 相位恢复

CPM 相位恢复模块利用“2P-Power”法恢复输入信号的载波频率。这种方法适用于应用 CPM、MSK、CPFSK、GMSK 等类型基带调制的系统。本模块可以和基带持续相位调制库中的其他模块共同使用。

如果将 CPM 中的调制系数表示为 $h=K/P$，那么 P 就是“2P-Power”所指的数。所谓“2P-Power”，就是如果在观测间隔之内的符号用 $x(1),x(2),x(3),\cdots,x(L)$ 表示，那么载波相位估计的结果为 $\frac{1}{2P}\arg\{\sum_{k=1}^{L}(x(k))^{2P}\}$。其中 arg 返回值在-180° 到 180° 之间。

“2P-Power”法假设系列连续符号的载波相位是个常量，并且返回这些系列的载波相位估计。参数“Observation interval”为模块认定具有相同载波相位的符号数，此项必须为输入向量长度的整数倍。输入信号代表符号速率的基带信号，因此为复值，并且要每个符号包含一个采样。

模块的输入必须是基于帧的列向量，或者是 double 或 single 类型的基于采样的标量。

模块的输出如下：

（1）标记为 Sig 的输出端输出输入信号的逆时针旋转，旋转量等于载波相位估计。因此 Sig 端输出的是输入信号的修正，和输入信号具有相同的采样时间和向量大小。

（2）标记为 Ph 的输出端输出为观测间隔内所有符号的载波相位估计，为一个标量。

CPM 相位恢复模块及其参数设定框如图 9-1 所示。

CPM 相位恢复模块包含下面两个参数项：

P：CPM（$h=K/P$）的调制系数表示为真分数时的分母 P。

Observation interval：设定假设载波相位为常量时的符号数。

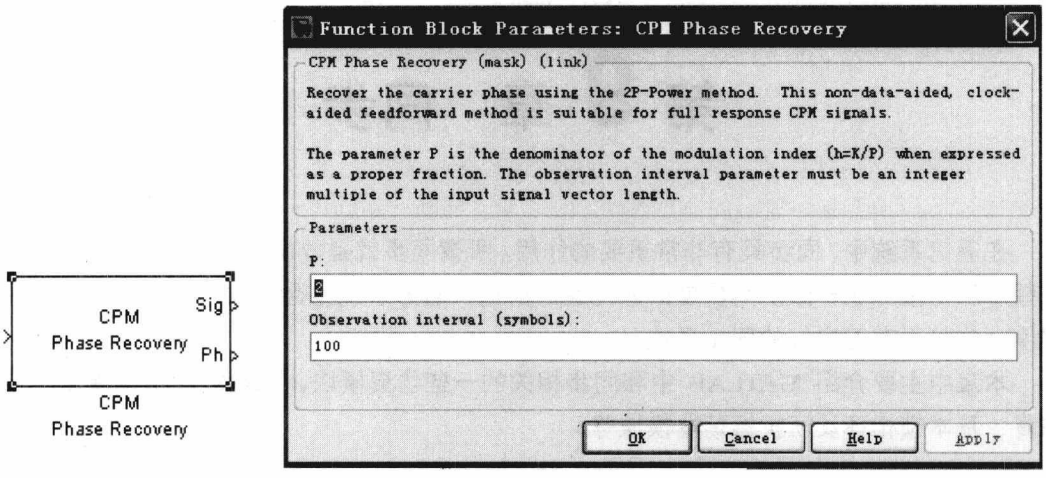


图 9-1　CPM 相位恢复模块及其参数设定框

9.1.2　M-PSK 相位恢复

M-PSK 相位恢复模块利用“M-Power”法恢复输入信号的载波频率。这种方法适用于应用 PSK、QAM 等调制的系统。调制的符号个数必须是偶数。

所谓“M-Power”，就是如果在观测间隔之内的符号用 $x(1),x(2),x(3),\cdots,x(L)$ 表示，那么载波相位估计的结果为 $\frac{1}{M}\arg\{\sum_{k=1}^{L}(x(k))^M\}$。其中 arg 返回值在-180° 到 180° 之间。

对于 PSK 信号，“M-ary number”项代表符号个数。对于 QAM 信号，“M-ary number”项恒定为 4，因为 4-Power 最适合于 QAM 信号。

“M-Power”法假设系列连续符号的载波相位是个常量，并且返回这些系列的载波相位估计。参数“Observation interval”为模块认定具有相同载波相位的符号数，此项必须为输入向量长度的整数倍。

模块的输入必须是基于帧的列向量，或者是 double 或 single 类型的基于采样的标量。

模块的输出如下：

（1）标记为 Sig 的输出端输出输入信号的逆时针旋转，旋转量等于载波相位估计。因此 Sig 端输出的是输入信号的修正，和输入信号具有相同的采样时间和向量大小。

（2）标记为 Ph 的输出端输出为观测间隔内所有符号的载波相位估计，为一个标量。

M-PSK 相位恢复模块及其参数设定框如图 9-2 所示。

M-PSK 相位恢复模块包含下面两个参数项：

M-ary number：PSK 信号时的星座图点数；传输 QAM 信号时为 4。此项必须设为偶数。

Observation interval：设定假设载波相位为常量时的符号数。

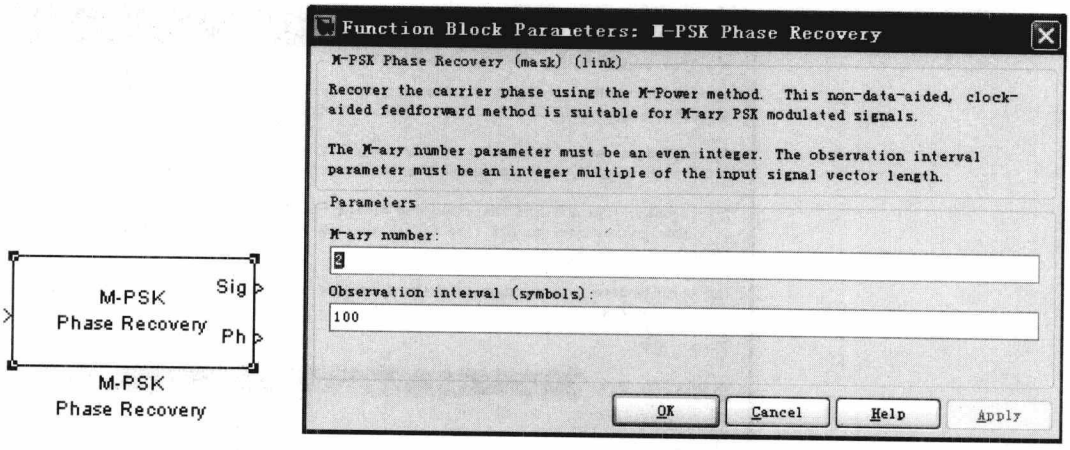


图 9-2 M-PSK 相位恢复模块及其参数设定框

9.2 定时恢复

我们知道，最佳接收机的一种普通的实现，是使用匹配滤波器和在匹配滤波器输出端的采样器。在所有的这些情况下，都假定接收机对采样瞬时具有完全的知识，并能够在这一时刻精确采样。在发射机和接收机之间完成这类同步的系统称为定时恢复。

MATLAB 中提供了多个定时恢复模块，包括 Early-Late Gate Timing Recovery、Gardner Timing Recovery、MSK-Type Signal Timing Recovery、Mueller-Muller Timing Recovery、Squaring Timing Recovery 等。本节以 MSK-Type Signal Timing Recovery 模块为例，做简单说明。

MSK-Type Signal Timing Recovery 模块利用四阶非线性方法来恢复符号时间相位。模块执行一个普通的非数据辅助反馈方法，它与载波相位恢复无关，但是要求优先补偿载波频率偏置。该模块适用于基带最小频率键控系统或高斯滤波最小频率键控系统。

模块的输入必须为标量或基于帧的列向量。模块的输入用 N 个采样代表一个符号。如果输入是基于帧的，那么其向量长度为 $N*R$，其中 R 为表示每帧符号数的正整数。如果输入是基于采样的，那么其采样时间是潜在符号周期的 $1/N$。

MSK-Type Signal Timing Recovery 模块及其参数设定框如图 9-3 所示。

MSK-Type Signal Timing Recovery 模块包含下面几个参数项：

Modulation type：设定系统中的调制方式，有 MSK 和 GMSK 两种方式，默认为 MSK。

Samples per symbol：该项参数 N 表示输入信号中每个符号中的采样数。本项必须大于 1。

Error update gain：该项必须为一个正实数，表示模块用于更新连续相位评估的步长。一般本项小于 $1/N$，表示是一个缓慢的相位变化。

Reset：重置项，设定在哪里以及在什么环境下模块重启相位评估过程。可选为 None、Every frame、On nonzero input via port 三种情况。如果选定为 On nonzero input via port，模块将会有另外一个标记为 Rst 的输入端。

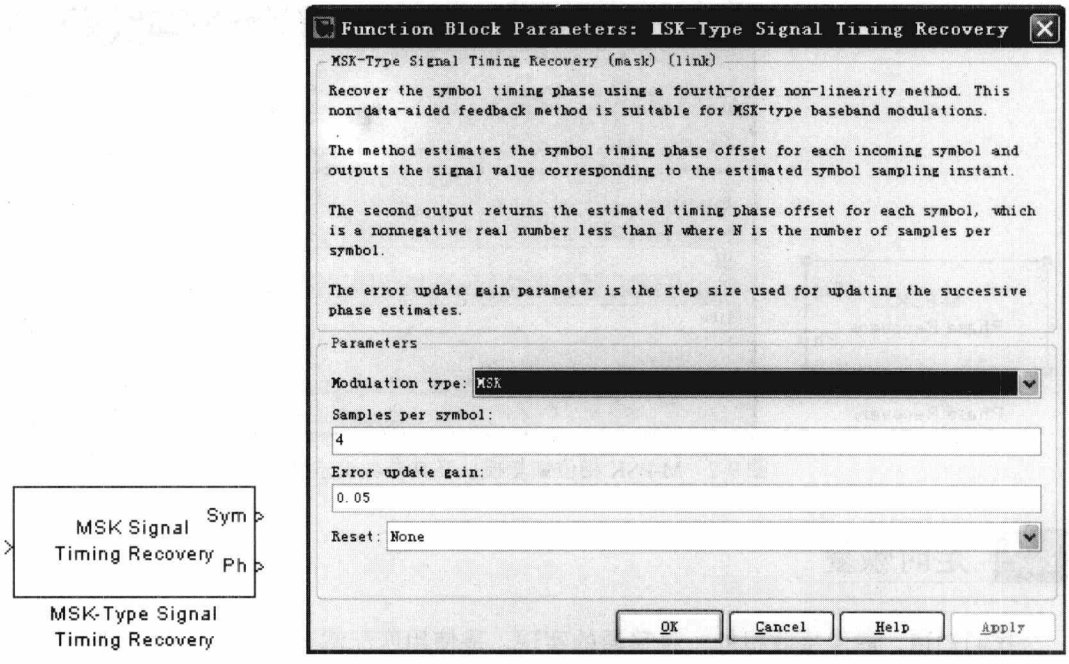


图 9-3　MSK-Type Signal Timing Recovery 模块及其参数设定框

9.3 基本锁相环及压控振荡器模块

9.3.1 基本锁相环

锁相环在同步中应用广泛，利用锁相环的跟踪能力，可以获得具有极小相位差的同步信号；利用锁相环的记忆功能，可以获得足够长的同步保持信号；利用锁相环的窄带滤波特性，可以滤除数据调制带来的白噪声及减小加性噪声的影响。MATLAB 中提供了多个锁相环模块，包括 Phase-Locked Loop、Linearized Baseband PLL、Charge Pump PLL、Baseband PLL 等，这里我们以 Phase-Locked Loop 为例做简单介绍。

Phase-Locked Loop 模块执行锁相环来恢复输入信号的相位。该模块能够自动地修正本地信号的相位来匹配输入信号的相位，最适用于窄带输入信号。

Phase-Locked Loop 模块包括三个部分：一个用于相位检测的乘法器、一个滤波器和一个压控振荡器。Phase-Locked Loop 模块及其参数设定框如图 9-4 所示。

Phase-Locked Loop 模块中包含下面几个参数项：

Lowpass filter numerator：低通滤波器转移函数的分子项，该项为一矢量，该矢量表示按照 S 的降序排列的多项式的系数。

Lowpass filter denominator：低通滤波器转移函数的分母项，该项为一矢量，该矢量表示按照 S 的降序排列的多项式的系数。

VCO input sensitivity (Hz/V)：本项用于衡量 VCO 的输入，进而衡量“VCO quiescent frequency”值的变化。单位为 Hz/V。

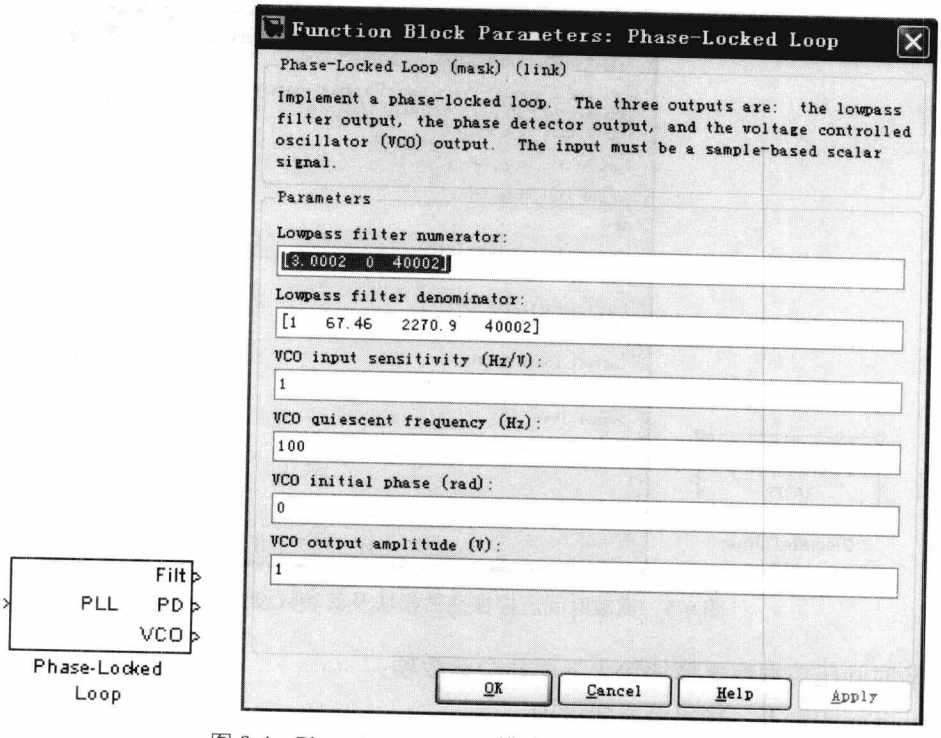


图 9-4 Phase-Locked Loop 模块及其参数设定框

VCO quiescent frequency (Hz)：电压为 0 时 VCO 信号的频率。本项应该与输入信号的载波频率相同。

VCO initial phase (rad)：本项表示 VCO 信号的初始相位。

VCO output amplitude：本项表示 VCO 信号的输出振幅。

9.3.2 压控振荡器

压控振荡器 VCO 是指输入信号的频率随着输入信号幅度的变化而发生相应变化的设备，它的工作原理可以用下式来表示：

$$y(t) = A_c \cos(2\pi f_c t + 2\pi k_c \int_0^t u(\tau) d\tau + \varphi)$$

其中 $u(t)$ 为输入信号，$y(t)$ 为输出信号，A_c 为信号幅度，f_c 为振荡频率，k_c 为输入信号灵敏度，φ 为初始相位。由于输入信号的频率取决于输入信号电压的变化，因此称为“压控振荡器”。

MATLAB 中提供了两种压控振荡器：离散时间压控振荡器和连续时间压控振荡器。两者的差别在于前者对输入信号 $u(t)$ 采用离散方式进行积分，而后者采用连续积分。下面分别对两个模块做简单介绍。

9.3.2.1 离散时间压控振荡器

离散时间压控振荡器模块及其参数设定框如图 9-5 所示。

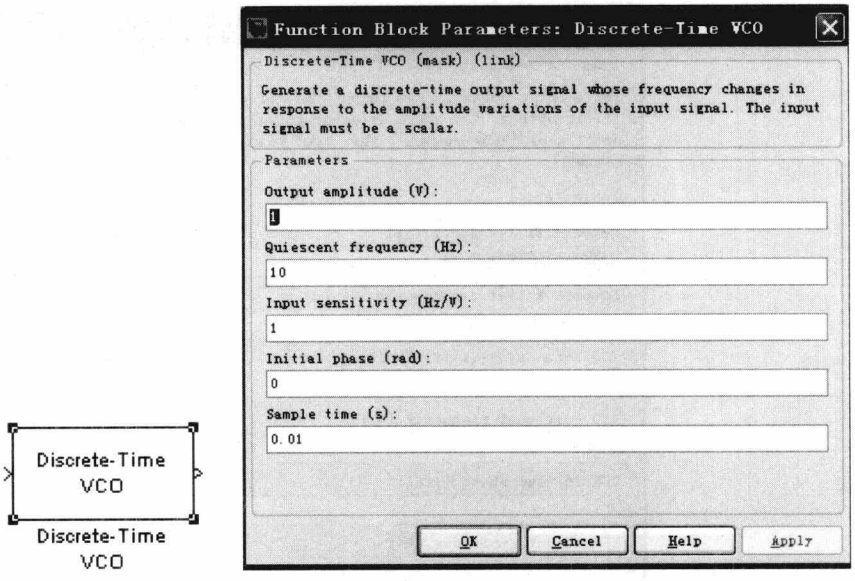


图 9-5　离散时间压控振荡器模块及其参数设定框

离散时间压控振荡器模块包含下面几个参数项：

Output amplitude：输出信号幅度项。

Quiescent frequency (Hz)：当输入信号为 0 时，离散时间压控振荡器的输出频率。

Input sensitivity：输入信号灵敏度。本项衡量输入电压，进而衡量“Quiescent frequency”值的变化。

Initial phase (rad)：离散时间压控振荡器的初始相位。

Sample time：采样时间项，表示离散积分的采样间隔。

9.3.2.2　连续时间压控振荡器

连续时间压控振荡器模块及其参数设定框如图 9-6 所示。

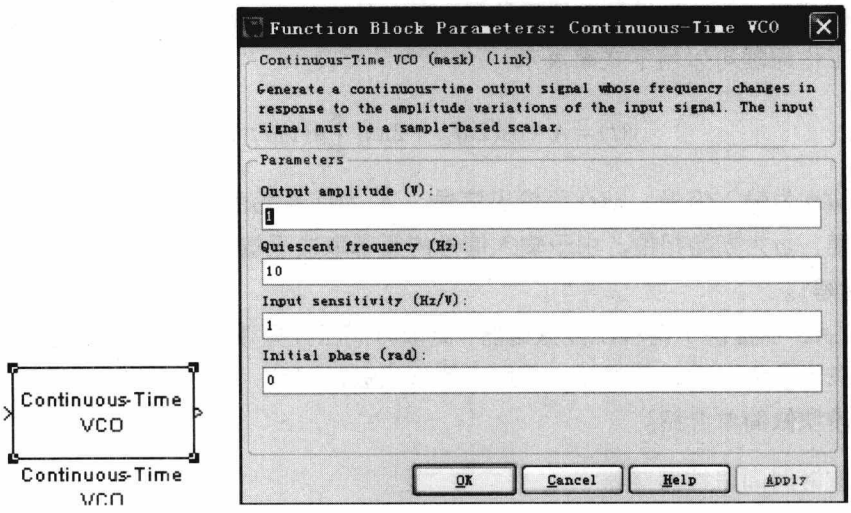


图 9-6　连续时间压控振荡器模块及其参数设定框

连续时间压控振荡器模块包含下面几个参数项：

Output amplitude：输出信号幅度项。

Quiescent frequency：当输入信号为 0 时，连续时间压控振荡器的输出频率。

Input sensitivity：输入信号灵敏度。本项衡量输入电压，进而衡量“Quiescent frequency”值的变化。

Initial phase (rad)：连续时间压控振荡器的初始相位。

9.4 本章小结

本章介绍了同步的一些相关知识，首先对 MATLAB 中提供的载波相位恢复、定时恢复等模块做了简单的说明，然后对常用的锁相环以及压控振荡器做了详细阐述，并以其中的几个模块为代表进行示范举例，以加深读者的理解程度，巩固学习效果。

笔记栏

Part 3

通信系统仿真综合实例篇

第 10 章　蓝牙跳频通信系统仿真设计

蓝牙是一种短距离无线通信的技术规范，其最初目标是取代现有的 PDA、移动电话、笔记本电脑等各类数字设备上的有线电缆连接。从目前应用来看，蓝牙体积小功耗低，已不仅局限于计算机外设，可集成至各种数字设备中，尤其是那些传输速率不高的移动设备和便携设备。

10.1 蓝牙技术概述

蓝牙技术工作在 2.4GHz 的 ISM 频段。虽然该频段为全球通用，但实际上频率和带宽在各国存在着一些差异。在美国，使用的带宽为 85.3MHz，在该频段里，以 1MHz 的带宽为间隔设立了 79 个射频跳频点。在日本、西班牙和法国，缩减了宽带，在频段里设立了 23 个射频跳频点，其宽带仍以 1MHz 为间隔，如表 10-1 所示。

表 10-1　可用射频信道

地　区	频率范围	射频信道
美国	2400 ~ 2485MHz	F=2402+kMHz　k=0,1,⋯,78
日本	2471 ~ 2479MHz	F=2473+kMHz　k=0,1,⋯,22
西班牙	2445 ~ 2475MHz	F=2449+kMHz　k=0,1,⋯,22
法国	2446.5 ~ 2483.5MHz	F=2454+kMHz　k=0,1,⋯,22

蓝牙采用电路交换和分组交换技术，支持异步数据信道、三路语音信道以及异步数据与同步语音同时传输的信道。每个语音信道数据速率为 64kbps，语音信号编码采用脉冲编码调制（PCM）或连续可变斜率增量调制（CVSD）技术。当采用对称信道传输数据时，最大传输速率为 721kbps，反向为 57.63kbps。当采用对称信号传输数据时，速率最高为 342.6kbps。其链路类型有两种：异步无连接（ACL）和同步面向连接（SCO）链路。在抗干扰方面，蓝牙采用跳频（Frequency Hoping）方式来进行扩频，它将 2.402 ~ 2.48GHz 频段分成 79 个频点，相邻频点间隔 1MHz。频点跳转序列是伪随机的，每秒钟频率改变 1600 次，每个频率持续 625μs。蓝牙技术具体指标和系统参数详见表 10-2。

表 10-2　蓝牙技术指标和系统参数

工作频段	ISM 频段：2.402GHz ~ 2.480GHz
双工方式	全双工、TDD 时分双工
业务类型	支持电路交换和分组交换业务
数据速率	1Mbps

续表

非同步信道速率	非对称连接 721kbps、57.6kbps，对称连接：432.6kbps
同步信道速率	64kbps
功率	FFC 要求小于 1mW，其他国家可扩展为 100mW
跳频频率数	79 个频点/MHz
跳频速率	1600time/s
工作模式	PARK/HOLD/SNIFF
数据连接方式	面向连接业务 SCO，无连接业务 ACL
纠错方式	1/3FEC，2/3FEC，ARQ
信道加密	采用 0 位、40 位、60 位加密字符
语音编码方式	连续可变斜率调制 CVSD
发射距离	10m，增加功率情况下可达 100m

10.2 蓝牙跳频系统各部分介绍

10.2.1 信号传输部分

蓝牙跳频系统信号传输主要包含两部分：信号序列产生和在跳频频率上映射该序列，如图 10-1 所示。

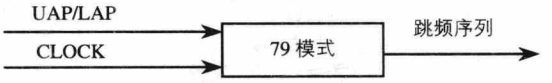


图 10-1　跳频映射

在输入端，输入的是本地时钟和当前地址。其中，使用时钟的 27 为 MSB。而在呼叫和查询状态下，将使用时钟的整个 28 位，在呼叫状态下，本地时钟将被修改为被叫单元对主单元的估算值。

地址输入位由 28 位组成，即整个 LAP 和 UAP 的 4 位 LSB。在连接状态中，可使用主单元地址，在呼叫状态下使用呼叫单元地址。而在查询状态下，使用和 GIAC 对应的 UAP/LAP。输出则构成一个伪随机序列。覆盖 79 跳还是 23 跳系统，取决于当前的状态。

对于 79 跳系统，将选择频率间隙为 64MHz 的 32 跳频段，并以随机次序访问这些频点一次。然后，选择一个不同的 32 跳频段，并以此类推。对于呼叫、呼叫扫描和呼叫应答状态，将使用同一个 32 跳频段。

信道被分为长度 625μs 的时隙。时隙依据主时钟来进行编号。编号区域为 $0 \sim 2^{27}-1$，循环周期为 $2^{27}-1$。在各时隙中，各单元和从单元都能够传输分组。

下面以跳频速率为 1600/s 的跳频系统为例，实现信号传输部分仿真，如图 10-2 所示。信号产生采用 Bernouli 随机信号生成模块生成帧采样率为 10、采样时间为 1.5e-6 的随机信号，具体参数设置如图 10-3 所示。信号经预处理在 1600/s 的跳频上进行映射。各模块参

数设置如图 10-4～图 10-6 所示。

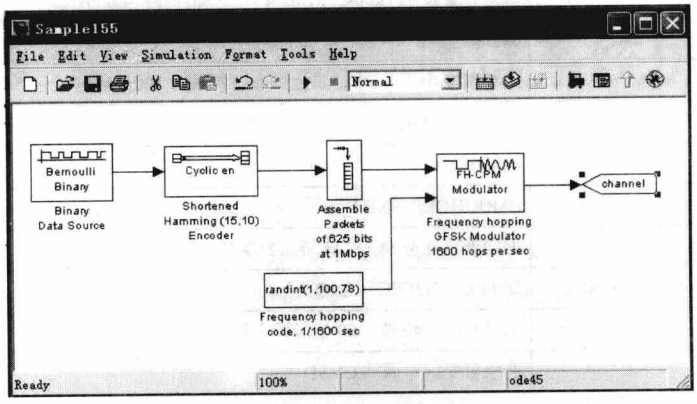


图 10-2　跳频系统传输部分仿真

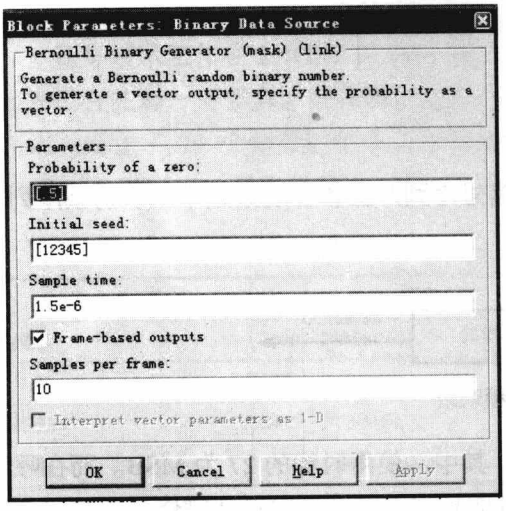


图 10-3　信号生成模块参数设置窗口

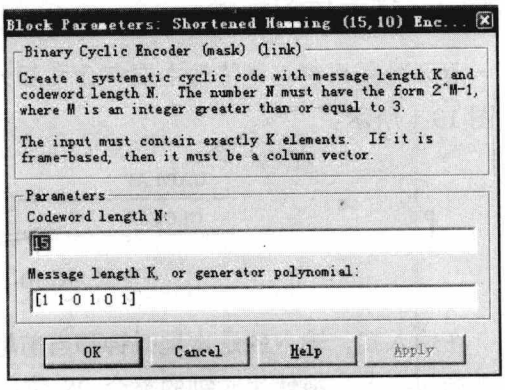


图 10-4　Shortend Hamming 模块参数设置窗口

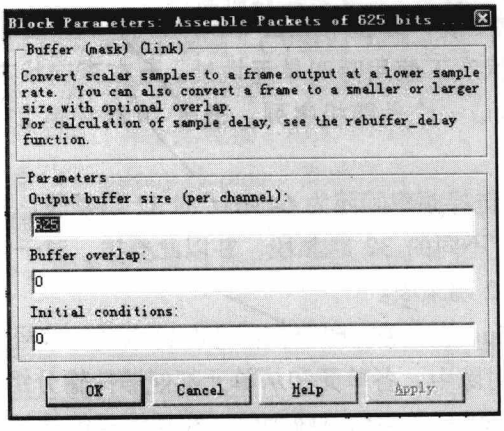


图 10-5　Buffer 模块参数设置窗口

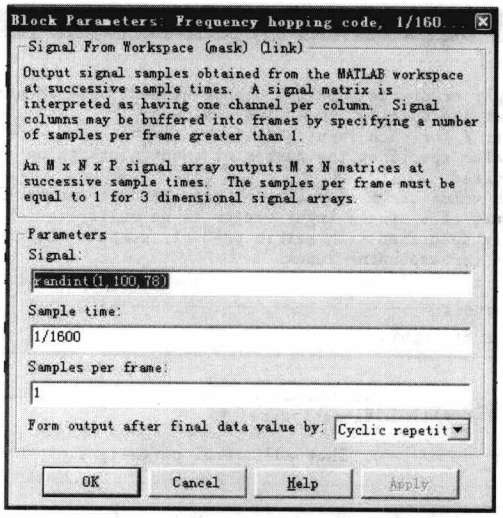


图 10-6　Singnal From Workspece 模块参数设置窗口

其中，跳频调制方式采用 FH-CPM 制式调制，该子系统内部结构如图 10-7 所示，输入 in1 将原始信号进行 CPM 调制（该模块参数设置见表 10-1）得到脉冲长度为 1 的 Binary 符号序列，在另一输入端将跳频速率为 1600/s 的跳频信号进行 M-FSK 调制，得到-39MHz～39MHz 的跳频序列，将二者相乘得到输出信号进入传输信道。各模块的参数设置如图 10-7～图 10-9 所示。两个调制模块的设置如表 10-3 与表 10-4 所示。

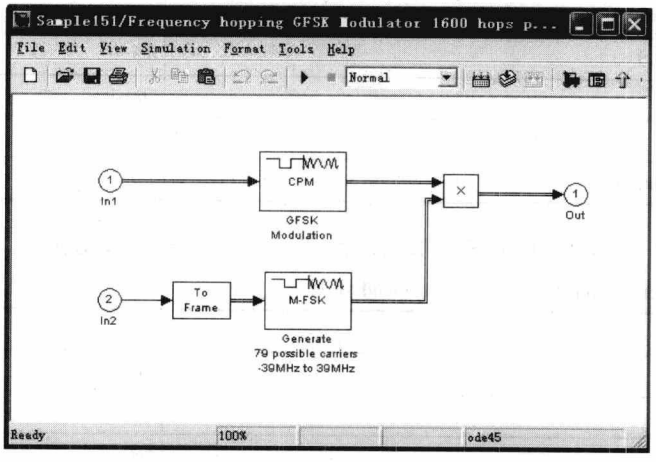


图 10-7　FH-CPM Modulaor 子系统

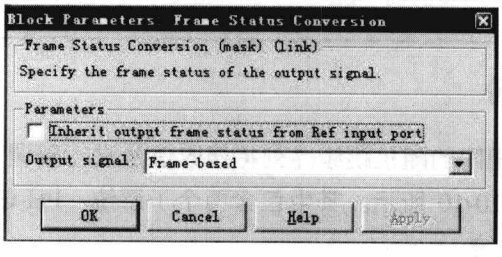


图 10-8　Fram Status Conversion 模块参数设置窗口

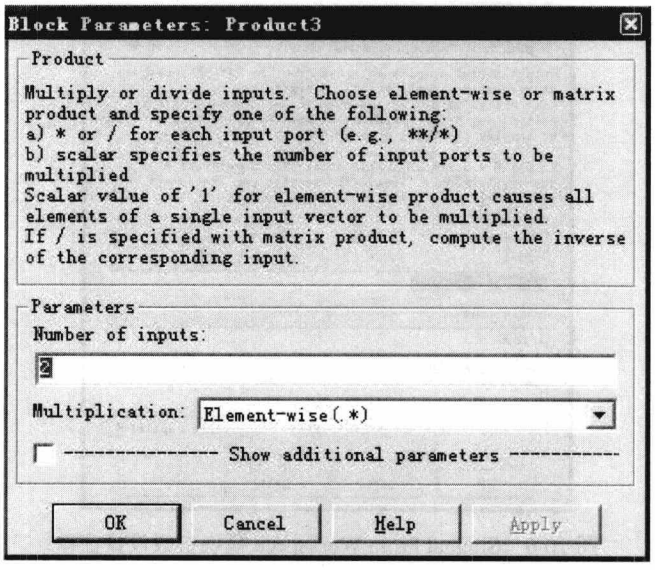


图 10-9　Product 模块参数设置窗口

表 10-3　M-FSK Modulation 模块参数设置

位置：Comunications Blockset/Digital Baseband Modulation/CPM

M-ary number （元数）	2
Input type（输入类型）	Bit
Symbol set ordering（符号序列）	Binary
Modulation Index （调制）	0.3
Frequence pulse shape	Gausian
BT product	0.5
Pulse length （symbol intervals）	1

表 10-4　M-FSK Modulator Baseband 模块参数设置

位置：Comunications Blockset/Digital Baseband Modulation/FM

M-ary number	79
Input type	Integer
Symbol set ordering	Binary
Frequence separation	1e6
Phase continuity	Continuous
Samples per symbol	6.25e4

10.2.2　信号接收部分

信号接收部分利用相同的随机跳频序列将接收信号进行解调，按照预处理的逆序进行解调，其仿真实现如图 10-10 所示。其中包含两个子系统：FH-CPM Demulator 子系统和 Dis-assemble Packet 子系统。

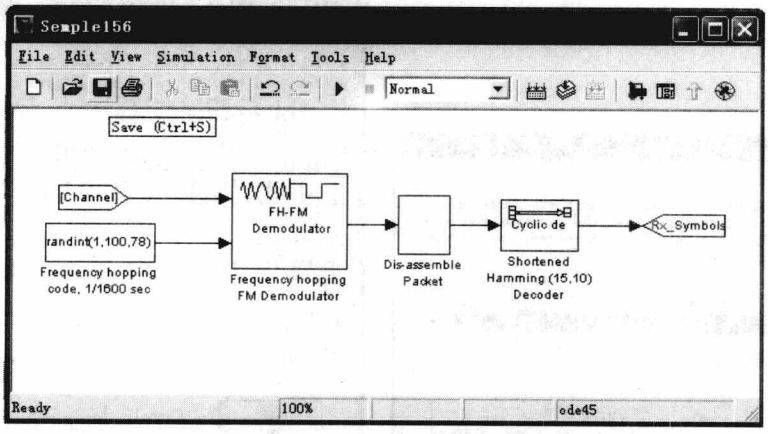


图 10-10　信号接收部分仿真

（1）FH-CPM Demulator 子系统

FH-CPM Demulator 子系统内部结构如图 10-11 所示，该子系统有两个输入端，in1 是经传输信道接收的扩频信号，in2 是随机序列产生器输入的随机跳频序列，它与发送端应保持同步，该序列经 M-FSK 调制与 in1 中的信号相乘再进行 M-FSK 解频，得到输出 out1。各模块的参数设置如图 10-12～图 10-16 所示。

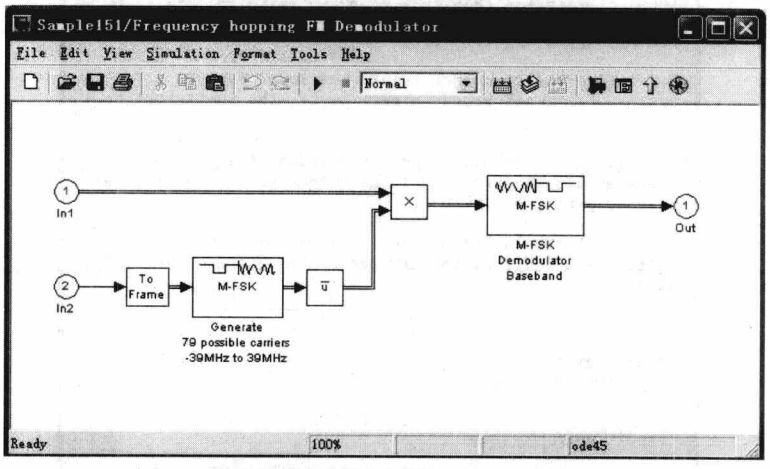


图 10-11　FH-CPM Demulator 子系统

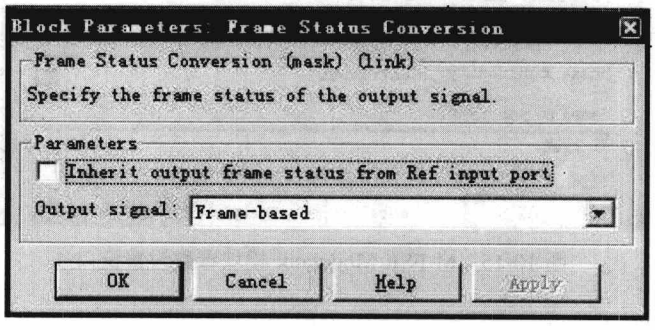


图 10-12　To Frame 模块参数设置窗口

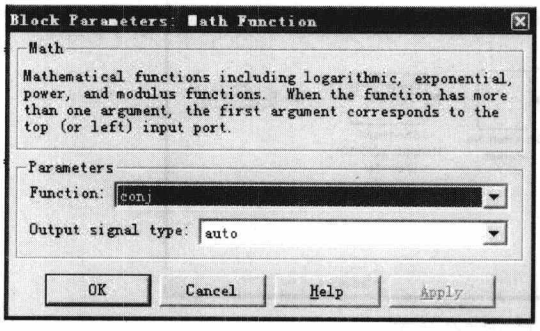


图 10-13　conj 模块参数设置窗口

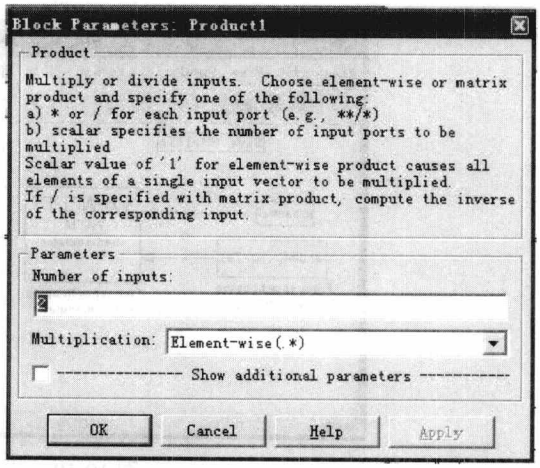


图 10-14　Product 模块参数设置窗口

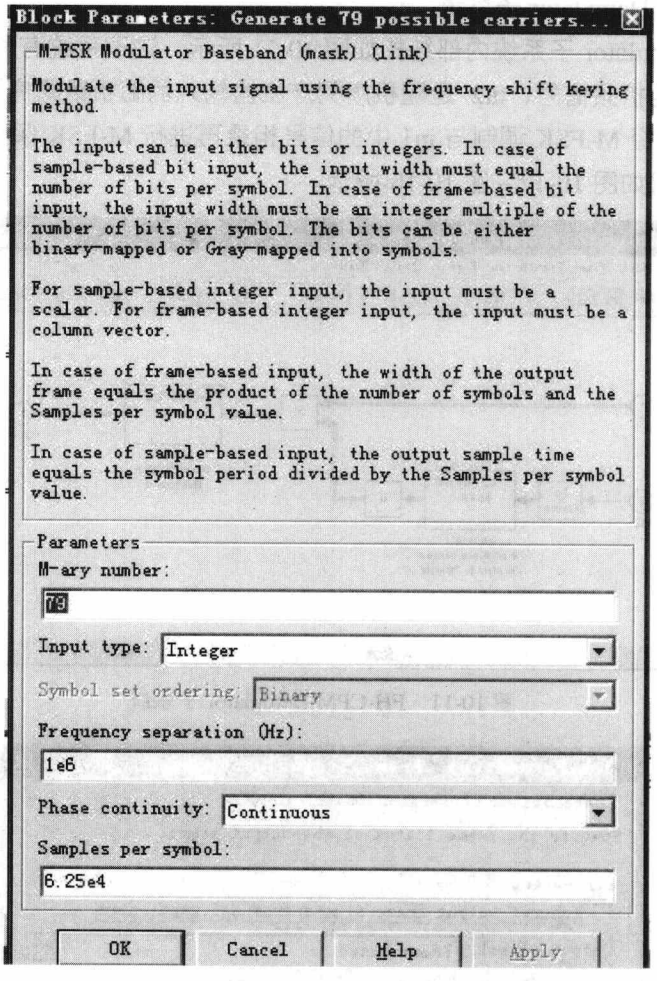


图 10-15　M-FSK Modulator 模块参数设置窗口

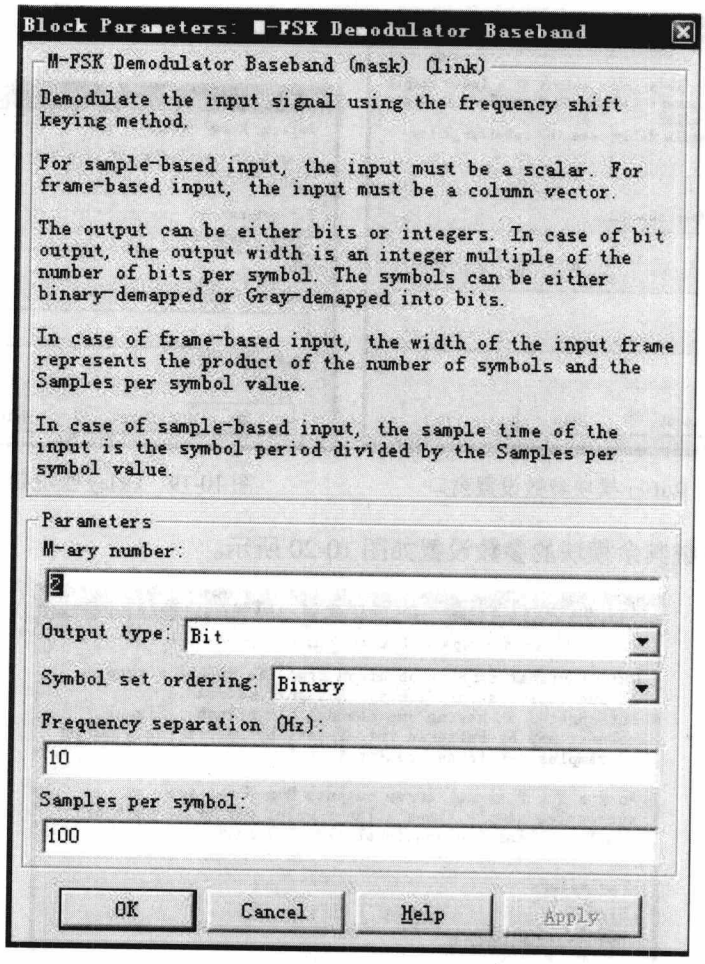


图 10-16　M-FSK Demodulator 模块参数设置窗口

（2）Dis-assemble Packet 子系统

Dis-assemble Packet 子系统内部结构如图 10-17 所示，由于经信道传输产生延迟，因此在 Dis-assemble Packet 中增加迟延 Integer Delay，采样延迟设置为 10。子系统各模块的参数设置如图 10-18、图 10-19 所示。

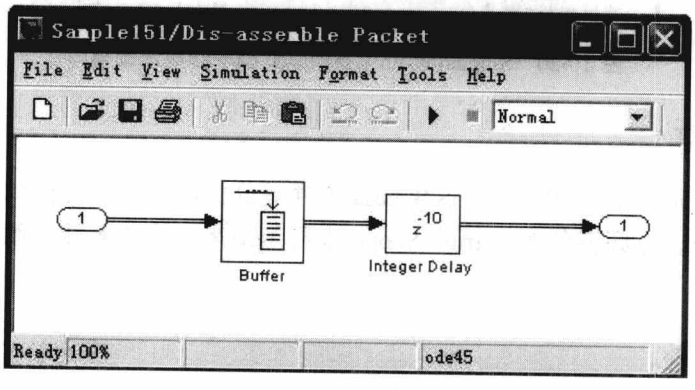


图 10-17　Dis-assemble Packet 子系统

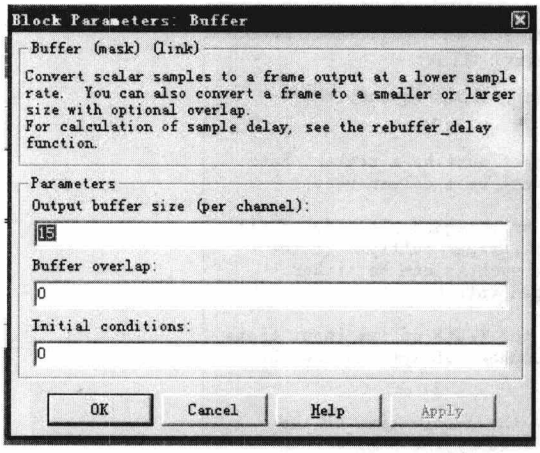


图 10-18　Buffer 模块参数设置窗口

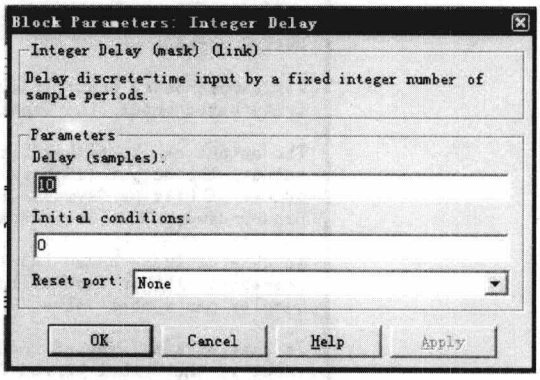


图 10-19　Delay 模块参数设置窗口

信号接收系统其余模块的参数设置如图 10-20 所示。

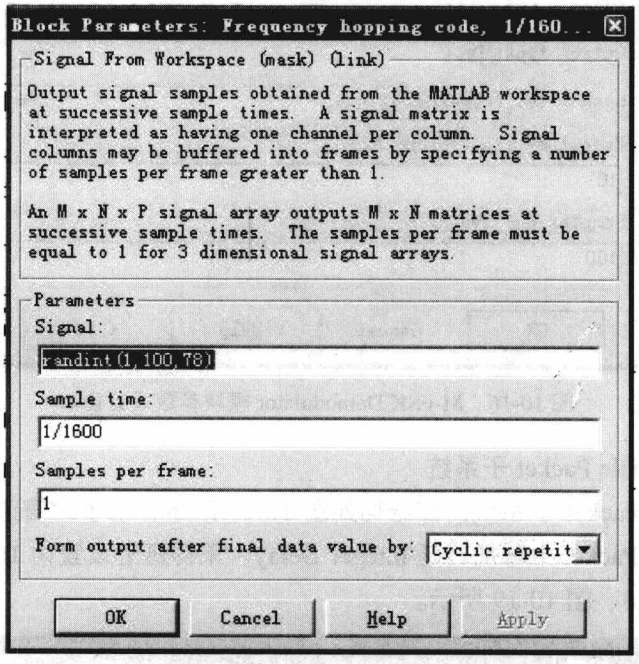


图 10-20　Singnal From Workspece 模块参数设置窗口

10.2.3　谱分析

为了在仿真系统中观察结果，这里设置了频谱分析模块，其结构如图 10-21 所示。信号通过选择器，在频谱仪 Spectrum Scope 中显示出来，模块参数如图 10-22、图 10-23 所示。

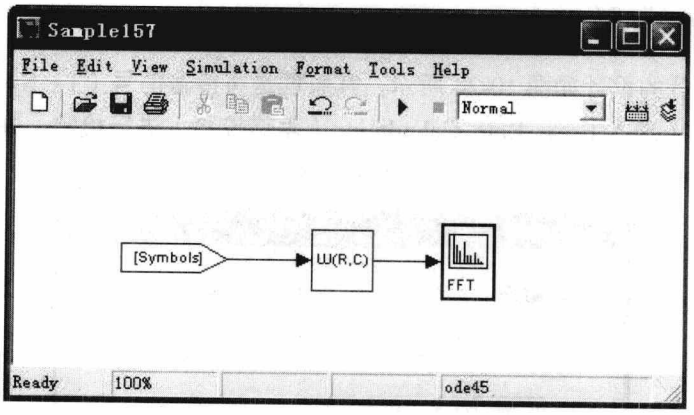


图 10-21　频谱分析模块

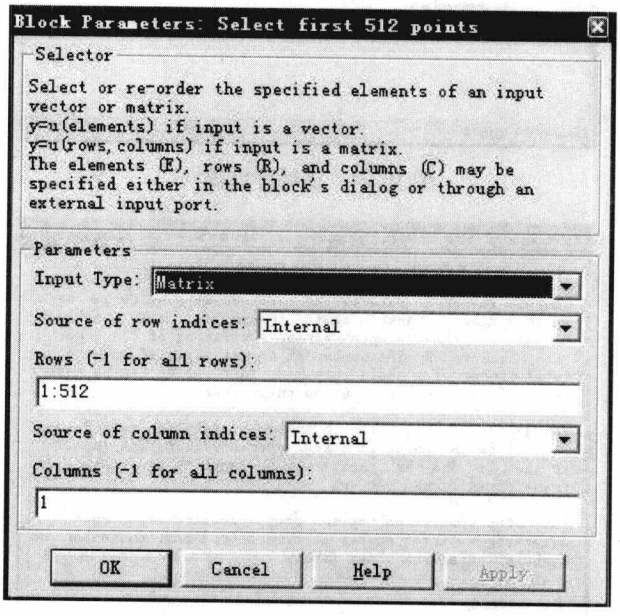


图 10-22　Selector 模块参数设置窗口

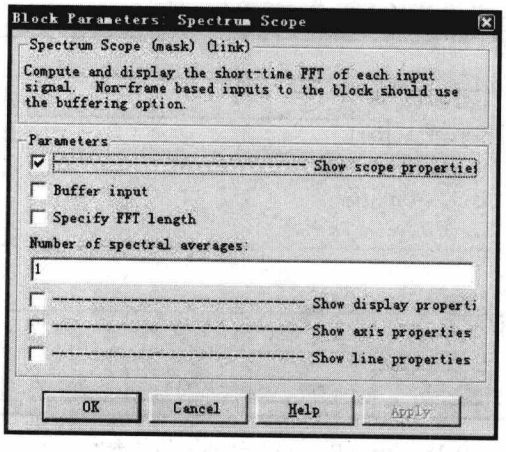


图 10-23　Spectrum Scope 模块参数设置窗口

10.2.4 误码分析部分

误码分析部分的设计如图 10-24 所示，其原理是将传输信号（Tx_Symbols）和接收信号（Rx_Symbols）送入 Error Rate Calculation（差错校验）进行检验，并将结果参数使用 Display 模块显示出来，模块参数设置如图 10-25、图 10-26 所示。

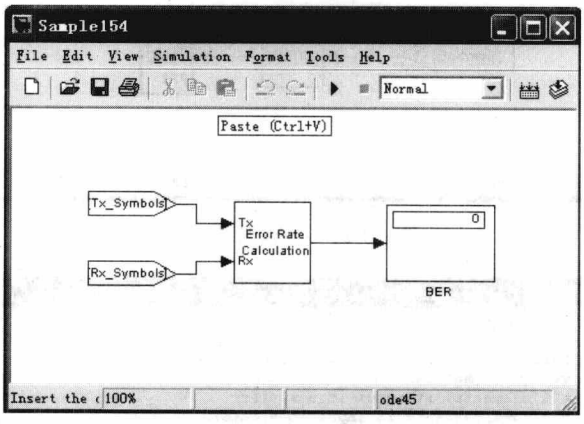


图 10-24 误码分析模块

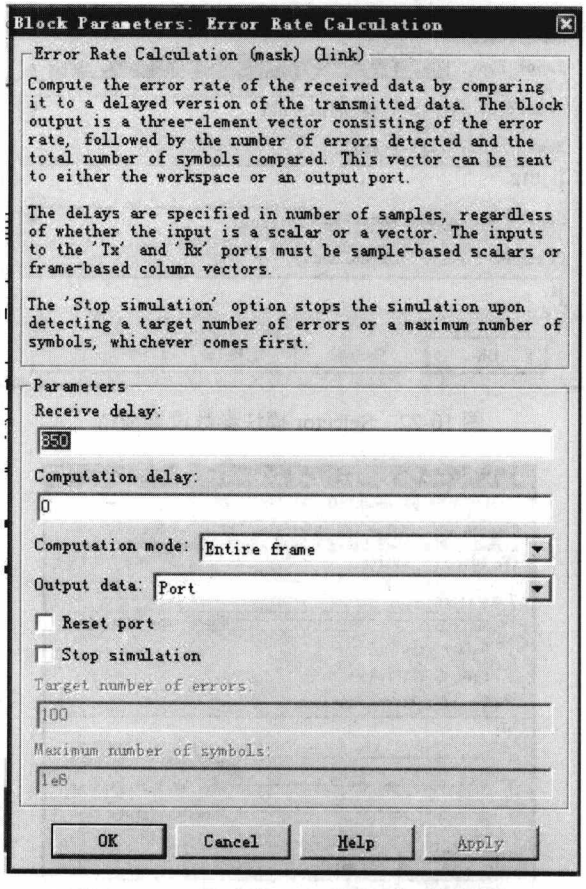


图 10-25 Error Rate Calculation 模块参数设置窗口

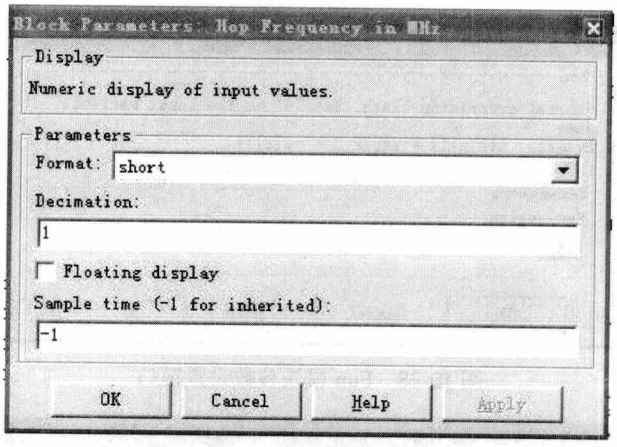


图 10-26　Display 模块参数设置窗口

10.3 蓝牙跳频系统的仿真模型

蓝牙跳频系统仿真模型如图 10-27 所示，传输信道采用加性高斯白噪声信道(AWGN)。

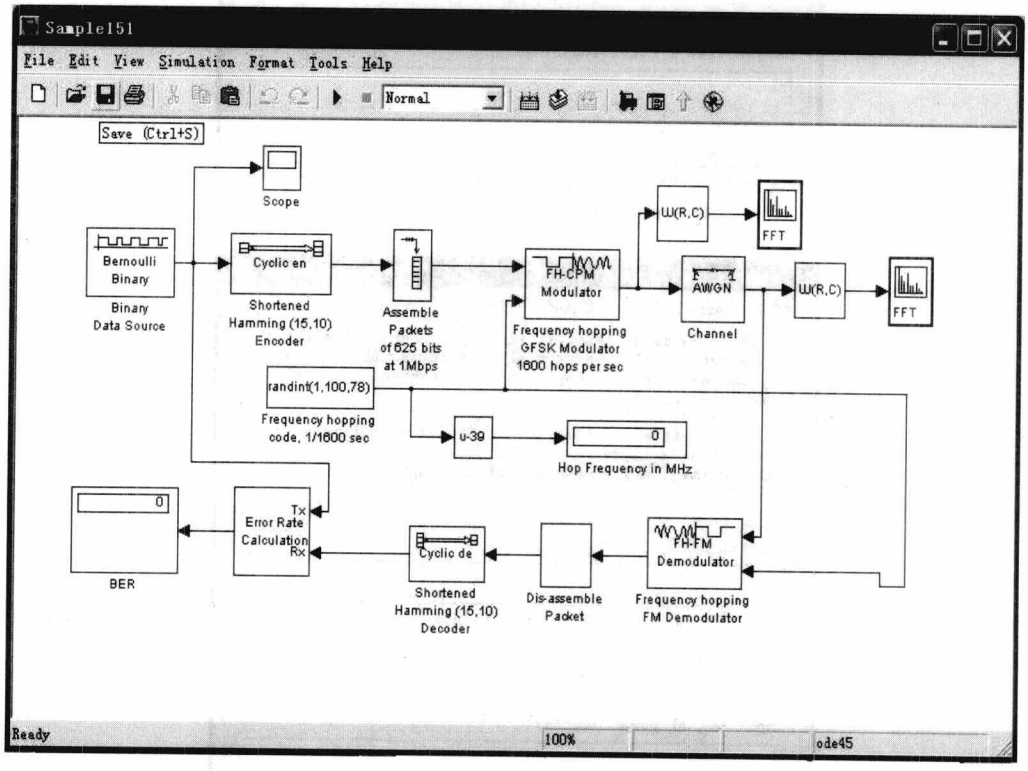


图 10-27　蓝牙跳频系统仿真模型

相关模块的参数设置如图 10-28 ~ 图 10-30 所示。

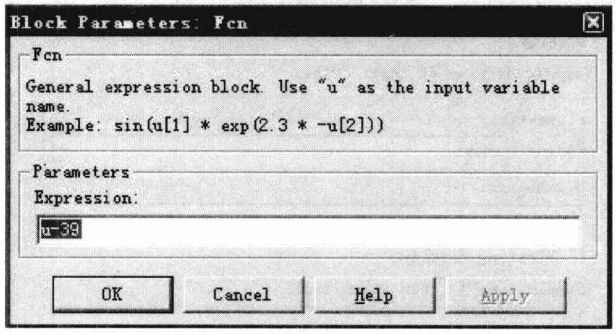


图 10-28　Fun 模块参数设置窗口

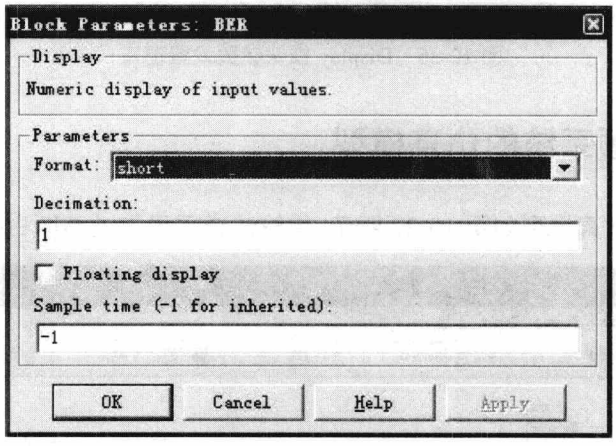


图 10-29　Display 模块参数设置窗口

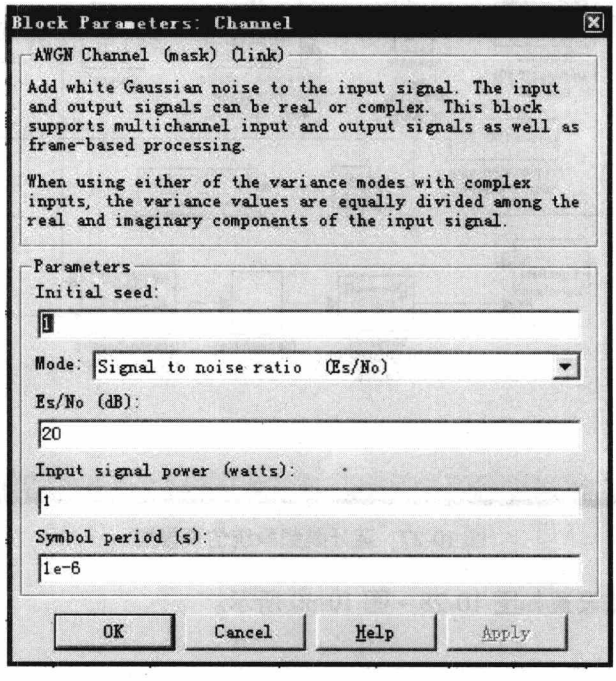


图 10-30　传输信道模块参数设置窗口

10.4 系统运行分析

在上面给出的仿真条件下，观察仿真运行情况。系统仿真结果频谱如图 10-31 所示，在频带范围-39MHz ~ 39MHz 内，产生 20 个分贝的随机跳频脉冲，经信道传输的信号频谱如图 10-32 所示，产生的误码率为 0。

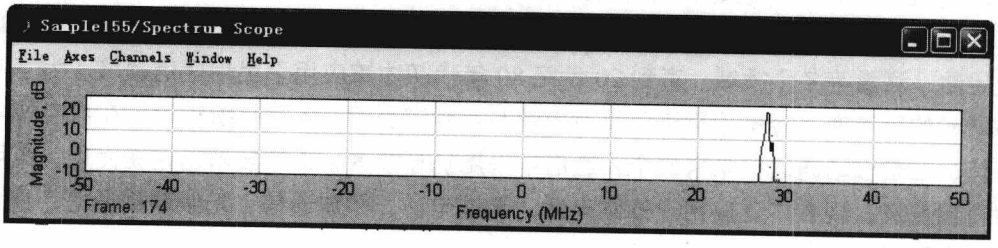


图 10-31　信号仿真频谱

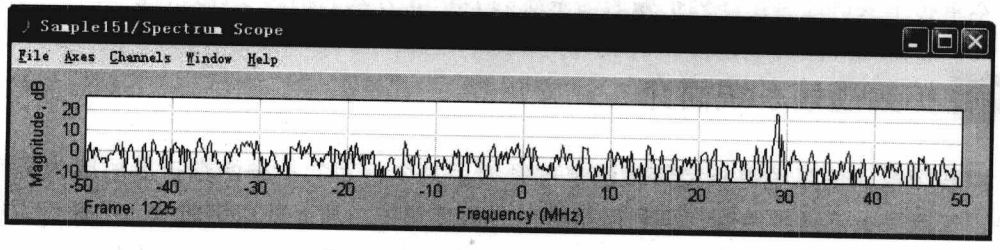


图 10-32　经信道传输的信号频谱

10.5 本章小结

本章首先对蓝牙跳频系统各部分的组成进行了介绍，然后给出了仿真模型，分析了系统运行情况。读者通过学习，可以了解和熟悉蓝牙跳频系统特点，掌握 MATLAB/Simulink 仿真设计的方法和相关原理。

第 11 章　直接序列扩频通信系统仿真设计

20 世纪 50 年代中期美国军方开始研究扩频通信，该技术在提高信号接收质量、抗干扰、保密性、增加系统容量等方面都有突出的优点，开始一直应用于军事通信、电子对抗以及导航、测量等各个领域。直到 20 世纪 80 年代初才被应用于民用通信领域。为了满足日益增长的民用通信容量的需求以及有效地利用频谱资源，各国都纷纷提出在数字蜂窝移动通信、卫星移动通信和未来的个人通信中采用扩频技术，扩频技术迅速在民用、商用领域中得到普及，现在广泛应用于蜂窝电话、无绳电话、微波通信、无线数据通信、遥测、监控、报警等系统中。

本章将主要讲述直接序列扩频通信系统的 DS/SS 方式的仿真设计过程。

11.1 扩频通信系统简介

扩展频谱通信系统是指待传输信息的频谱用某个特定的扩频函数扩展后成为宽频带信号，然后送入信道中传输，再利用相应手段将其解扩，从而获取传输信息的通信系统。它在传输同样信息时所需的射频带宽，远比大家熟悉的各种调制方式要求的带宽要宽得多。扩频前的信息码元带宽远小于扩频后的扩频码序列的带宽，信息已不再是决定调制信号带宽的一个重要因素，其调制信号的带宽主要由扩频函数来决定。一般常用的扩频函数是伪随机编码信号。与常规的通信系统相比，扩频系统具有很强的抗人为干扰、抗窄带干扰、抗多径的能力，此外还具有信息隐蔽多址保密通信等优点。

11.1.1　技术理论基础

扩展频谱技术的理论基础可以用香农信道容量公式来描述：

$$C = W\log_2(1+S/N) \qquad (式\ 11\text{-}1)$$

上式表明，在高斯信道中当传输系统的信号噪声功率比下降时，可以用增加系统传输带宽 W 的办法来保持信道容量 C 不变。对于任意给定的信号噪声比，可以用增大传输带宽来获得较低的信息差错率。可见，扩展频谱技术正是利用这一原理，用高速率的扩频码来达到扩展待传输的数字信息带宽的目的。扩频通信系统的带宽比常规通信系统大几百倍至几千倍，所以在相同的信噪比条件下，具有较强的抗噪声干扰能力。

其次，香农又指出：在高斯噪声的干扰下，在平均功率受限的信道上，实现有效和可靠通信的最佳信号是具有白噪声统计特性的信号。这是因为高斯白噪声信号具有理想的自相关特性，其功率谱为：

$$S(\omega)=N_0/2 \qquad -\infty<\omega<\infty \tag{式 11-2}$$

其自相关函数为

$$R(\tau)=\frac{1}{2\pi}\int_{-\infty}^{\infty}S(\omega)\mathrm{e}^{\mathrm{j}\omega\tau}\mathrm{d}\omega=\frac{N_0}{2}\delta(\tau) \tag{式 11-3}$$

其中 $\omega=2\pi f$ ，τ 为时延，$\delta(\tau)=\begin{cases}1, & \tau=0\\ 0, & \tau\neq 0\end{cases}$

白噪声的自相关函数具有 $\delta(\tau)$ 函数的特点，说明它具有尖锐的自相关特性。但是对于白噪声信号的产生、加工和复制，至今存在着许多技术上的困难。然而，有许多易于产生又便于加工和控制的伪随机噪声序列，它们的统计特性逼近于高斯白噪声信号的统计特性。

设某种伪随机序列周期为 p，且码元都是二元域 $\{1,-1\}$ 上的元，假设一个周期为 p，码元为 x 的伪随机序列 X 的归一化自相关函数为：

$$R_x(j)=\begin{cases}\dfrac{1}{p}\displaystyle\sum_{i=1}^{p}x_i\bullet x_{i+j}=1 & \text{当}j=0\text{时（模}p\text{）}\\ \dfrac{1}{p}\displaystyle\sum_{i=1}^{p}x_i\bullet x_{i+j}=-\dfrac{1}{p} & \text{当}j\neq 0\text{时（模}p\text{）}\end{cases} \tag{式 11-4}$$

其中 $j=0,1,2,\cdots,p-1$。当伪随机序列码长 p 取足够长或趋于无穷时，上式可以简化为：

$$R_x(j)=\begin{cases}1 & \text{当}j=0\text{时（模}p\text{）}\\ 0 & \text{当}j\neq 0\text{时（模}p\text{）}\end{cases} \tag{式 11-5}$$

可以看到式（11-5）中当 p 足够长或趋于无穷时，该伪随机序列和白噪声信号有类似的统计特性，也就是逼近于高斯信道要求的最佳信号形式。可见用伪随机码扩展待传基带信号的扩展频谱通信系统，优于常规通信体制。

扩频通信的工作原理如图 11-1 所示。

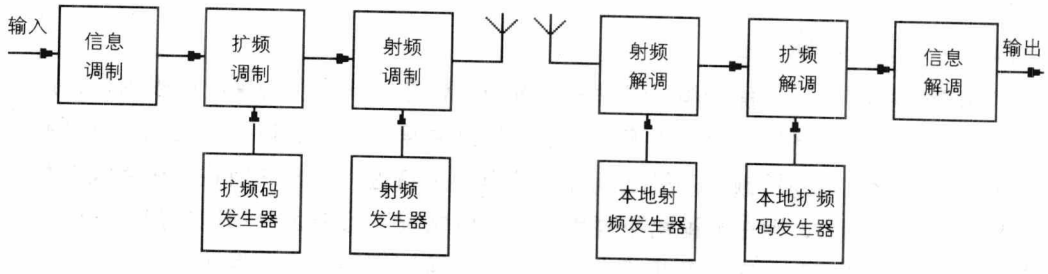


图 11-1　扩频通信的工作原理

在发端输入的信息先经信息调制变成数字信号，然后由扩频码发生器产生的扩频码序列去调制数字信号以展宽信号的频谱。展宽后的信号再调制到射频发送出去。在接收端收到的宽带射频信号，变频至中频，然后由本地产生的与发端相同的扩频序列进行相关解扩，再经信息解调恢复成原始信息输出。

11.1.2 系统主要特点

由于扩频通信能大大扩展信号的频谱，发送端用扩频码序列进行扩频调制，以及在接收端用相关解调技术，使其具有许多窄带通信难以替代的优良性能，能在民用后，迅速推广到各种公用和专用通信网络之中，主要有以下几项特点：

（1）易于重复使用频率，提高了无线频谱利用率

无线频谱十分宝贵，虽然从长波到微波都得到了开发利用，仍然满足不了社会的需求。在窄带通信中，主要依靠波道划分来防止信道之间发生干扰。为此，世界各国都设立了频率管理机构，用户只能使用申请获准的频率。扩频通信发送功率极低，采用了相关接收技术，且可工作在信道噪声和热噪声背景中，易于在同一地区重复使用同一频率，也可与现今各种窄道通信共享同一频率资源。所以，在美国及世界绝大多数国家，扩频通信无须申请频率，任何个人与单位都可以无执照使用。

（2）抗干扰性强，误码率低

扩频通信在空间传输时所占用的带宽相对较宽，而接收端又采用相关检测的办法来解扩，使有用宽带信息信号恢复成窄带信号，而把非所需信号扩展成宽带信号，然后通过窄带滤波技术提取有用的信号。这样，对于各种干扰信号，因其在接收端的非相关性，解扩后窄带信号中只有很微弱的成分，信噪比很高，因此抗干扰性强。当处理增益 G_p=35dB 时，抗干扰容限 M_j=22dB，即在负信噪声比（–22dB）条件下，可以将信号从噪声中提取出来。在目前商用的通信系统中，扩频通信是唯一能够工作在负信噪比条件下的通信方式。

（3）隐蔽性好，对各种窄带通信系统的干扰很小

由于扩频信号在相对较宽的频带上被扩展了，单位频带内的功率很小，信号湮没在噪声里，一般不容易被发现，而想进一步检测信号的参数如伪随机编码序列就更加困难，因此说其隐蔽性好。再者，由于扩频信号具有很低的功率谱密度，它对目前使用的各种窄带通信系统的干扰很小。

（4）可以实现码分多址

扩频通信提高了抗干扰性能，但付出了占用频带宽的代价。如果让许多用户共用这一宽频带，则可大大提高频带的利用率。由于在扩频通信中存在扩频码序列的扩频调制，充分利用各种不同码型的扩频码序列之间优良的自相关特性和互相关特性，在接收端利用相关检测技术进行解扩，则在分配给不同用户码型的情况下可以区分不同用户的信号，提取出有用信号。这样一来，在一宽频带上许多对用户可以同时通话而互不干扰。

（5）抗多径干扰

在无线通信的各个频段，长期以来，多径干扰始终是一个难以解决的问题。在以往的窄带通信中，采用以下两种方法来提高抗多径干扰的能力：一是把最强的有用信号分离出来，排除其他路径的干扰信号，即采用分集接收技术；二是设法把不同路径来的不同延迟、不同相位的信号在接收端从时域上对齐相加，合并成较强的有用信号，即采用梳状滤波器的方法。

这两种技术在扩频通信中都易于实现。利用扩频码的自相关特性，在接收端从多径信号中提取和分离出最强的有用信号，或把多个路径来的同一码序列的波形相加合成，这相当于梳状滤波器的作用。另外，在采用频率跳变扩频调制方式的扩频系统中，由于用多个频率的信号传送同一个信息，实际上起到了频率分集的作用。

（6）能精确地定时和测距

我们知道电磁波在空间的传播速度是固定不变的光速，人们自然会想到如果能够精确测量电磁波在两个物体之间的传播时间，也就等于测量两个物体之间的距离。在扩频通信中如果扩展频谱很宽，则意味着所采用的扩频码速率很高，每个码片占用的时间就很短。当发射出去的扩频信号在被测量物体反射回来后，在接收端解调出扩频码序列，然后比较收发两个码序列相位之差，就可以精确测出扩频信号往返的时间差，从而算出两者之间的距离。测量的精度决定于码片的宽度，也就是扩展频谱的宽度。码片越窄，扩展的频谱越宽，精度越高。

（7）适合数字话音和数据传输，以及开展多种通信业务

扩频通信一般都采用数字通信、码分多址技术，适用于计算机网络，适合于数据和图像传输。

（8）安装简便，易于维护

扩频通信设备是高度集成，采用了现代电子科技的尖端技术，因此，十分可靠、小巧，大量运用后成本低，安装便捷，易于推广应用。

11.1.3　系统基本类型

扩频系统包括以下几种扩频方式：

- 直接序列扩频，记为 DS（Direct Sequence）：这种方式运用最为普遍，成为行业领域研究的热点，下文将有详细介绍。
- 跳频扩频，记为 FH（Frequency Hopping）：频率跳变系统有称为“多频、码选、频移键控”系统，主要由码产生器和频率合成器两部分组成。
- 跳时扩频，记为 TH（Time Hopping）：跳时扩频系统主要通过扩频码控制发射机的通断，可以减少时分复用系统之间的干扰。
- 上述各方式的混合形式。上述三种基本扩频系统各有优缺点，单独使用一种系统有时难以满足要求，将以上集中扩频方法结合就构成了混合扩频系统，常见的有 FH/DS、TH/DS、FH/TH 等。

11.2　直接序列扩频通信系统原理

直接序列扩频通信系统（Direct Sequence Spread System，DSSS）简称直扩系统，是目前应用最广泛的扩频系统。早期的一些军事领域的研究开发，例如全球定位系统（Global Position System，GPS）、航天飞机通信用的跟踪和数据中继卫星系统等都是直接序列扩频

系统应用的实例。

直接序列扩频系统是将要发送的信息用伪随机序列扩展到一个很宽的频带上去。在接收端，用于发射端扩展用的相同的伪随机序列对接收到的扩频信号进行相关处理，恢复出原来的信息。由于干扰信息与伪随机序列不相关，在接收端被扩展，使落入信号频带内的干扰信号功率大大降低，从而提高了系统的输出信噪比，达到抗干扰的目的。

11.2.1 系统结构

图 11-2 为直扩系统发送框图，信源输出信号 $a(t)$ 是码元持续时间为 T_a 的信息流，伪随机码发生器产生的伪随机码为 $c(t)$，每一随机码元宽度为 T_c。将信码 $a(t)$ 和伪随机码 $c(t)$ 进行模二加，产生一速率与伪随机码速率相同的扩频序列，然后再用扩频序列去调制载波，这样就得到了已扩频调制的射频信号。在接收端，接收到的扩频信号经高放和混频后，用与发端速率及相位都相同的伪随机序列对中频的扩频调制信号进行相关解扩，将信号的频带恢复为信号序列 $a(t)$ 的频带，为中频调制信号，然后再进行解调，恢复出所传输的信息 $a(t)$，从而完成信息的传输，直扩系统接收框图如图 11-3 所示。对于干扰信号和噪声而言，由于与伪随机序列不相关，在相关解扩器的作用下，相当于进行了一次扩频。干扰信号和噪声频谱被扩展后，其谱密度降低，这样就大大降低了进入信号通频频带内的干扰功率，使解调器的输入信噪比和输入信干比提高，从而大大提高了系统的抗干扰能力。

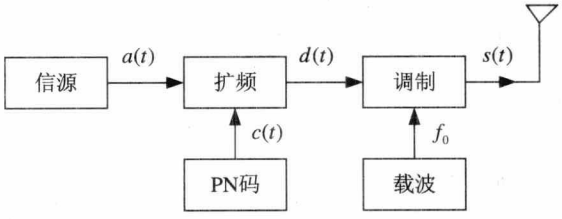


图 11-2 直扩系统发送框图

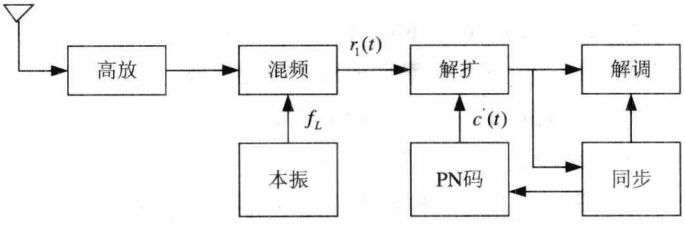


图 11-3 直扩系统接收框图

11.2.2 信号分析

信号源产生的信号 $a(t)$ 为信息流，码元速率为 R_a，码元宽度为 T_a，$T_a = 1/R_a$，其中

$$a(t) = \sum_{n=-\infty}^{\infty} a_n g_a(t - nT_a) \qquad (式\ 11\text{-}6)$$

其中 a_n 为信息码，取值为+1 或者-1，$g_a(t)$ 为门函数。

伪随机序列发生器产生的信号伪随机序列 $c(t)$，速率为 R_c，码片宽度为 T_c，$T_c = 1/R_c$。

$$c(t) = \sum_{n=-\infty}^{\infty} c_n g_c(t - nT_a) \qquad (式\ 11\text{-}7)$$

其中 c_n 为信息码，取值为+1 或者-1，$g_c(t)$ 为门函数。

扩频过程实质上是信息流 $a(t)$ 与伪随机序列 $c(t)$ 的模二加或者相乘的过程。伪随机码速率 R_c 比信息速率 R_a 大得多，一般 R_c / R_a 的比值为整数，且 $R_c / R_a >> 1$，所以扩展后序列的速率仍为伪随机码的速率 R_c。扩展后的序列 $d(t)$ 为

$$d(t) = a(t)c(t) = \sum_{n=-\infty}^{\infty} d_n g_c(t - nT_c) \qquad (式\ 11\text{-}8)$$

其中 $d_n = \begin{cases} +1 & a_n = c_n \\ -1 & a_n \neq c_n \end{cases}$，$(n-1)T_c \leqslant t \leqslant nT_c$。

用此扩展后的序列去调制载波，将信号搬移到载频上去，用于直扩系统的调制。一般来说，大多数数字调制方式均可，但应视具体情况，根据系统的性能要求来确定，用得较多的调制方式有 BPSK、MSK、QPSK 等。这里分析采用 BPSK 调制，其他的调制方式可以类似讨论。调制后的信号 $s(t)$ 为

$$s(t) = d(t)\cos\omega_0 t = a(t)c(t)\cos\omega_0 t \qquad (式\ 11\text{-}9)$$

其中，ω_0 为载波频率。

接收端天线上感应的信号经高放的选择放大和混频后，得到包括以下几部分的信号：有用信号 $s_I(t)$、信道噪声 $n_I(t)$、干扰信号 $J_I(t)$ 和其他网的扩频信号 $s_J(t)$ 等，即接收到的信号（经混频后）为

$$r_I(t) = s_I(t) + n_I(t) + J_I(t) + s_J(t) \qquad (式\ 11\text{-}10)$$

接收端的伪随机码产生器产生的伪随机序列与发端产生的伪随机序列相同，但起始时间或者初始相位可能不同，为 $c'(t)$。解扩的过程与扩频过程相同，用本地的伪随机序列 $c'(t)$ 与接收到的信号相乘，相乘后为

$$\begin{aligned} \hat{r}(t) &= r_I(t)c'(t) \\ &= s'_I(t) + n_I(t)c'(t) + J_I(t)c'(t) + s_J(t)c'(t) \\ &= s'_I(t) + n'_I(t) + J'_I(t) + s'_J(t) \end{aligned} \qquad (式\ 11\text{-}11)$$

下面分别对式 11-11 中的四个分量进行分析，首先看信号分量 $s'_I(t)$：

$$s'_I(t) = s_I(t)c'(t) = a(t)c(t)c'(t)\cos\omega_0 t \qquad (式\ 11\text{-}12)$$

若本地产生的伪随机序列 $c'(t)$ 与发端产生的伪随机序列 $c(t)$ 同步时，有 $c'(t) = c(t)$，则 $c'(t)c(t) = 1$。这样，信号分量 $s'_I(t)$ 为

$$s'_I(t) = a(t)\cos\omega_0 t \qquad (式\ 11\text{-}13)$$

后面所接滤波器的频带正好能让信号通过，因此可以进入解调器进行解调，将有用信号解调出来。

对噪声分量 $n_I'(t)$、干扰分量 $J_I'(t)$ 和不同网干扰 $s_J'(t)$，经解扩处理后，被大大削弱。这里的 $n_I'(t)$ 一般为高斯带限白噪声，因而用 $c'(t)$ 处理后，谱密度基本不变，但相对带宽改变，因此噪声功率降低。$J_I'(t)$ 分量是人为干扰信号引起的。由于与伪随机码不相关，因此相乘过程相当于频谱扩展过程，将干扰信号功率分散到很宽的频带上，谱密度降低。相乘器后接的滤波器只能让有用信号通过，因而能够进入到解调器输入端的干扰功率只能是与信号频带相同的那一部分。解扩前后的频带相差甚大，因而解扩后干扰功率大大降低，提高了解调器输入端的信噪比，从而提高了系统抗干扰的能力。对于不同网的信号 $s_J'(t)$，由于不同网采用不同的伪随机码序列，序列之间只有很低的互相关性，相当于再次扩展，从而降低了不同网信号的干扰。

11.2.3 处理增益和干扰容限

1. 处理增益 G_p

在扩频系统中，传输信号经过扩频和解扩的处理，系统的抗干扰性能得到提高，这种扩频处理得到的好处，称之为扩频系统的处理增益，其定义为接收相关处理器输出与输入信噪比的比值，即

$$G_p = \frac{\text{输出信噪比}}{\text{输入信噪比}} = \frac{S_0 / N_0}{S_i / N_i} \qquad (式\ 11\text{-}14)$$

一般用分贝表示为

$$G_p = 10\lg\frac{S_0 / N_0}{S_i / N_i}(\text{dB}) \qquad (式\ 11\text{-}15)$$

经过分析可知

$$G_p = \frac{B_c}{B_a} = \frac{R_c}{R_a} \qquad (式\ 11\text{-}16)$$

由此可见，直扩系统的处理增益为扩频信号射频带宽与传输的信息带宽的比值，或为伪随机码速率 R_c 与信息速率 R_a 的比值，即直扩信号的扩频倍数。在一般情况下，发送信息的带宽是不变的，要提高扩频系统的抗干扰能力，就应该提高扩频系统的处理增益，即提高扩频用的伪随机码的速率。

当处理增益提高到一定程度时，不能再靠提高伪随机码速率的方法来提高系统的处理增益，而应考虑用别的办法，比如可以降低信息速率，而且可能更为有效一些。

2. 干扰容限 M_j

所谓干扰容限，是指在保证系统工作正常的条件下，接收机能够承受的干扰信号比有用信号高出的分贝数。用 M_j 表示

$$M_j = G_p - \left[L_s + \left(\frac{S}{N}\right)\right](\text{dB}) \qquad (式\ 11\text{-}17)$$

式中 L_s 为系统内部损耗；S/N 为系统正常工作时要求的最小输出信噪比；G_p 即系统

的处理增益。

干扰容限直接反映了扩频系统接收机可能抵抗的极限干扰强度，即只有当干扰机的干扰功率超过干扰容限时，才能对扩频系统形成干扰。因而干扰容限往往比处理增益更确切地反映系统的抗干扰能力。

3. 码分多址

多址接入或多址联接是多用户无线通信网中按用户地址进行连接的问题。在移动通信系统中，基站覆盖区内存在许多移动台，移动台必须能识别基站发射的信号中哪一个是发给自己的信号，基站也必须从众多的移动台发射信号中识别出每一个移动台所发射的信号。由此可见，多址（接入）技术在数字蜂窝移动通信中占有重要的地位。

多址接入方式的数学基础是信号的正交分割原理。无线电信号可以表达为时间、频率和码型的函数，即可写作：

$$s(c,f,t)=c(t)s(f,t) \qquad (式\ 11\text{-}18)$$

其中 $c(t)$ 是码型函数，$s(f,t)$ 是时间 t 和频率 f 的函数。

当以传输信号的载波频率不同来区分信道建立多址接入时，称为频分多址方式（FDMA）；当以传输信号存在的时间不同来区分信道建立多址接入时，称为时分多址方式（TDMA）；当以传输信号的码型不同来区分信道建立多址接入时，称为码分多址方式（CDMA）。图 11-4 分别给出了 N 个信道的 FDMA、TDMA 和 CDMA 的示意图。

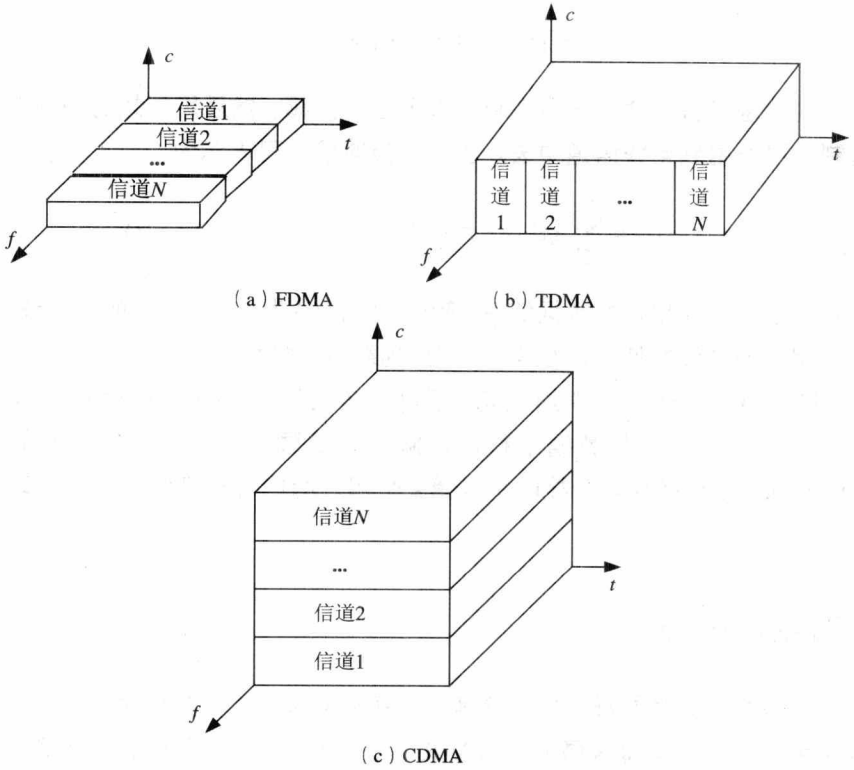


图 11-4　FDMA、TDMA 和 CDMA 示意图

CDMA（Code Division Multiple Access）是使用一组正交（或准正交）的伪随机噪声（PN）序列（简称伪随机码）通过相关处理来实现多个用户共享空间传输的频率资源和同时入网接续的功能，即码分多址。CDMA 只能由扩频调制来完成，而扩频调制并不意味着 CDMA。

CDMA 的特点包括以下几点：

- CDMA 系统的许多用户共享同一频率。
- 通信容量大。
- 由于信号被扩展在一较宽频谱上，因而可以减小多径衰落。
- 在 CDMA 系统中，信道数据速率很高。
- 平滑的软切换和有效的宏分集。
- 低信号功率谱密度。

图 11-5 为 DS-CDMA 系统的原理图。

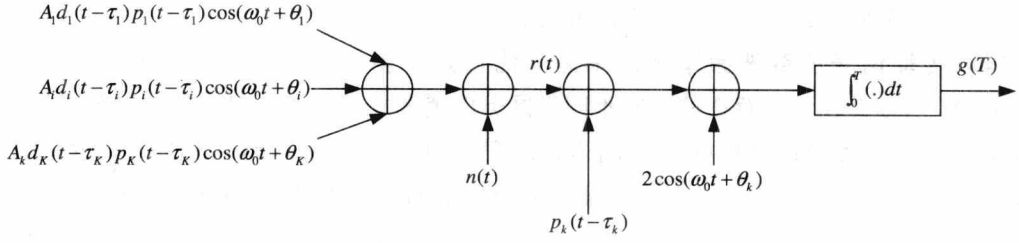


图 11-5　DS-CDMA 系统原理图

各用户的数据字符先使用各自的特征波形进行扩频，然后用载频的调频信号对扩频信号进行调制，再发射到无线信道中去。因此，接收信号可表示为：

$$r(t)=\sum_{i=1}^{K}A_id_i(t-\tau_i)p_i(t-\tau_i)\cos(\omega_0t+\theta_i)+n(t) \qquad （式 11-19）$$

其中 $d_i(t)$ 为第 i 个用户的数据信息，取值 ±1；$p_i(t)$ 为第 i 个用户的扩频波形，取值 ±1；A_i 为第 i 个用户的载波幅值；θ_i 为第 i 个用户的载波的随机相位，在 $[0,2\pi]$ 内均匀分布；τ_i 为第 i 个用户的随机时延，在 $[0,T]$ 内均匀分布；T 为码元间隔；$n(t)$ 为加性高斯白噪声。

假定针对第 k 个用户的数据字符进行解扩和解调。为此，接收信号 $r(t)$ 分别乘以 $p_k(t-\tau_k)$ 和 $2\cos(\omega_0t+\theta_k)$。令接收机与第 k 个用户的信号正确同步，则可以求得相关接收机的输出 $g(T)$。

11.3 伪随机序列

在扩频系统中，信号频谱的扩展是通过扩频码实现的。扩频系统的性能与扩频码的性能有很大关系，对扩频码通常提出下列要求：易于产生；具有随机性；扩频码应该具有尽可能长的周期，使干扰者难以从扩频码的一小段中重建整个码序列；扩频码应该具有双值

自相关函数和良好的互相关特性，以利于接收时的捕获和跟踪，以及多用户检测。

从理论上说，用纯随机序列去扩展频谱是最理想的，例如高斯白噪声，但在接收机中为了解扩应当有一个同发送端扩频码同步的副本，因此实际上只能用伪随机或伪噪声序列作为扩频码。伪随机序列具有貌似噪声的性质，但它又是周期性有规律的，易于产生和处理。

扩频码中应用最广的是 m 序列，又称最大长度序列，其他还有 Gold 序列、L 序列和霍尔序列等。

11.3.1　m 序列

m 序列是最长线性移位寄存器序列，是由移位寄存器加反馈后形成的。m 序列是伪随机序列中最重要的一种，其易于实现，具有优良的自相关特性，在直扩通信系统中用于扩展要传递的信号。它的产生如图 11-6 所示。

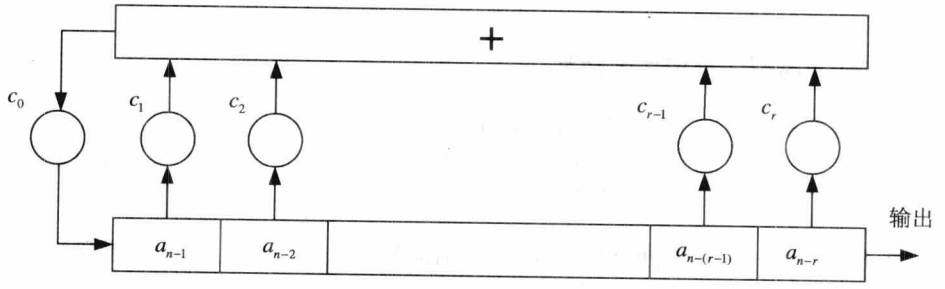


图 11-6　反馈移位寄存器结构

图 11-6 中 $a_{n-i}(i=1,2,3,\cdots,r)$ 为移位寄存器中每位寄存器的状态，取值为 1 或者 0；$c_i(i=1,2,3,\cdots,r)$ 为对应于第 i 位寄存器的反馈系数。当 $c_i=0$ 时，表示无反馈，将反馈线断开。当 $c_i=1$ 时，表示有反馈，将反馈线连接起来，注意在此结构中必须保证 $c_0=c_r=1$。

在把 m 序列作为扩频码时，每一个码元有一定的宽度，设为 T_c。下面推导码元宽度为 T_c 时扩频码的相关函数。设二元脉冲序列码元宽度为 T_c，幅度为+1 和-1 的概率各为 1/2，所有脉冲幅度值相互独立。脉冲波形起始时间为 T，T 在 0 到 T 之间均匀分布。任取两个时刻 t_1 和 t_2，且 $0<t_1<t_2<T_c$。当延时 $|\tau|=kT_c(k=0,1,2,...,p-1)$ 时，m 序列的自相关函数为：

$$r(t)=\begin{cases}1 & \tau=0\\ -\dfrac{1}{p} & \tau\neq 0\end{cases} \qquad |\tau|=kT_c \quad (k=0,1,2,...,p-1) \qquad (式\ 11\text{-}20)$$

当 $|\tau|<T_c$ 时，t_1 和 t_2 可以在同一脉冲区间内，也可以在相邻两个脉冲区间内。设 A 代表 t_1、t_2 落在同一脉冲区间内的事件，即 (t_1,t_2) 内没有脉冲沿的事件。A 代表 t_1、t_2 落在相邻两脉冲区间内的事件。则事件 A 发生的概率为：

$$P(A)=P(T<t_1或T>t_2)=\frac{t_1}{T_c}+\frac{T_c-t_2}{T_c}$$
$$=1-\frac{t_2-t_1}{T_c} \quad (式\ 11\text{-}21)$$

A 不发生的概率为：

$$P(\bar{A})=P(t_1<T<t_2)=\frac{t_2-t_1}{T_c} \quad (式\ 11\text{-}22)$$

若 $t_2<t_1$，式 11-21 和式 11-22 变为：

$$P(A)=1-\frac{t_1-t_2}{T_c} \quad (式\ 11\text{-}23)$$

$$P(\bar{A})=\frac{t_1-t_2}{T_c} \quad (式\ 11\text{-}24)$$

随机二元脉冲序列的自相关函数为：

$$R_x=E[x(t_1)x(t_2)]$$
$$=\sum_{x_1}\sum_{x_2}x_1x_2P(x_1x_2) \quad (式\ 11\text{-}25)$$

事件 A 出现时，x_1x_2 的取值有(1,1)、(-1,-1)、(1,-1)、(-1,1)四种情况。这样，$P(x_1x_2|\bar{A})=1/4$。

由全概率公式：

$$P(x_1x_2)=P(x_1x_2|A)P(A)+P(x_1x_2|\bar{A})P(\bar{A})$$
$$=\frac{1}{2}P(A)+\frac{1}{4}P(\bar{A}) \quad (式\ 11\text{-}26)$$

由式 11-26 和式 11-23 可知随机二元脉冲序列的自相关函数为：

$$R_x=\sum_{x_1}\sum_{x_2}x_1x_2P(x_1x_2)$$
$$=P(A) \quad (式\ 11\text{-}27)$$
$$=1-\frac{|t_1-t_2|}{T_c}$$

令 $\tau=t_1-t_2$，上式变为：

$$R_x=1-\frac{|\tau|}{T_c} \quad (式\ 11\text{-}28)$$

当 $|t_1-t_2|>T_c$ 时，$P(A)=0$，$R_x(\tau)=0$，于是可以得到周期为 p 的 m 序列扩频码的自相关函数为：

$$R_x(\tau)=\begin{cases}1-\dfrac{|\tau|}{T_c}\left[1+\dfrac{1}{p}\right] & |\tau|\leqslant T_c \\ -\dfrac{1}{p} & T_c\leqslant|\tau|\leqslant\dfrac{pT_c}{2}\end{cases} \qquad (式\ 11\text{-}29)$$

由上式可以看出 m 序列扩频码的自相关函数是周期为 pT_c 的周期三角函数，据此可以推导 m 序列扩频码的功率密度谱。

由于功率密度谱和相关函数是一对傅里叶变换，m 序列扩频码的自相关函数可以看成是高度为 $p(1+p)$ 的周期三角形脉冲序列减去一个幅度为 $1/p$ 的直流分量。设周期三角形脉冲的傅里叶变换为 $G_1(f)$，幅度为 $1/p$ 的直流分量的傅里叶变换为 $G_2(f)$，显然，m 序列扩频码的功率密度谱 $G(f)$ 等于：

$$G(f)=G_1(f)-G_2(f) \qquad (式\ 11\text{-}30)$$

三角脉冲的傅里叶变换是抽样函数的平方，周期三角脉冲的傅里叶变换是以抽样函数平方为包络的离散谱线，相邻谱线的间隔为 $1/pT_c$。这样，$G_1(f)$ 可以表示为：

$$G_1(f)=\frac{p+1}{p^2}\left(\frac{\sin\pi fT_c}{\pi fT_c}\right)^2\sum_{-\infty}^{\infty}\delta\left(f-\frac{k}{pT_c}\right) \qquad (式\ 11\text{-}31)$$

直流分量的傅里叶变换为：

$$G_2(f)=\frac{1}{p^2}\delta(f) \qquad (式\ 11\text{-}32)$$

所以 m 序列扩频码的功率密度谱 $G(f)$ 为：

$$G(f)=\frac{1}{p^2}\delta(f)+\frac{p+1}{p^2}\left(\frac{\sin\pi fT_c}{\pi fT_c}\right)^2\sum_{-\infty}^{\infty}\delta\left(f-\frac{k}{pT_c}\right) \qquad (式\ 11\text{-}33)$$

由式 11-33 可以看出序列扩频码功率谱具有如下特性：

m 序列扩频码功率谱是周期函数，它的自相关函数也是同周期的周期函数，相应的功率密度谱就是线状谱，相邻的谱线间隔为 pT_c。由于序列波形是幅度恒定的矩形波，因而具有恒定的功率，除零频率分量外，各谱线强度与序列长度 p 成反比。零频率分量的强度为 $1/p^2$，与序列的长度 p^2 成反比。m 序列扩频码功率谱密度的包络由码元宽度 T_c 决定，而与序列周期 pT_c 无关。这说明传输 m 序列随机信号的频带宽度决定于码元宽度 T_c。

现将 m 序列的特点总结如下：

（1）均衡性

以 N 为周期的序列中包含 2^{N-1} 个“1”和 $2^{N-1}-1$ 个“0”，“1”和“0”的个数基本相同（“1”比“0”的数目多一个）。这一性质很重要，因为码序列或码序列调制信号中的直流分量将决定于码的均衡性，另外由于载波抑制度决定于调制信号的对称性，当用一个码序列去调制载波时，0-1 均衡性将限制可达到的载波抑制度。

（2）游程分布

把一个序列中取值相同的那些相继元素合称为一个游程。一般来说，在 m 序列中，长度为 1 的游程占游程总数的 1/2；长度为 2 的游程占游程总数 1/4；长度为 3 的占 1/8…，即长度为 k 的游程占游程总数的 2^{-k}，其中 $1<k<(r-2)$。

（3）移位相加性

一个序列与其经过 m 次延迟移位产生的另一序列模 2 加，得到的仍然是原序列的 m 次延迟移位序列。

（4）周期性

m 序列的周期为 $N=2^r-1$，r 为反馈移位寄存器的级数。

（5）伪随机性

m 序列的各个性质与随机序列的基本性质很相似，所以通常认为 m 序列属于一种常用的伪随机序列。

（6）相关特性

m 序列的自相关函数只有两种取值（1 和 $-1/N$）。

11.3.2 Gold 序列

虽然 m 序列具有很好的伪随机性和相关特性，且使用简单，但是 m 序列的个数相对较少，很难满足作为系统地址码的要求。Gold 码继承了 m 序列的许多优点，而可用码的个数又远大于 m 序列，是一种良好的码型。

Gold 码是 R. Gold 提出的用优选对的复合码，所谓 m 序列优选对，是指在 m 序列集中，其互相关函数最大值的绝对值小于某个值的两条 m 序列。而 Gold 码是由两个长度相同、速率相同、但码字不同的 m 序列优选对模 2 加后得到的，具有良好的自相关性及互相关特性。因为一对序列优选对可产生 2^r+1 条 Gold 码，所以 Gold 码的条数远远大于 m 序列。

Gold 码具有三值互相关函数，其值为：

$$-\frac{1}{p}t(r),-\frac{1}{p},\frac{1}{p}[t(r)-2] \qquad \text{（式 11-34）}$$

这里，

$$p=2^r-1,\quad t(r)=\begin{cases}1+2^{0.5(r+1)}, & r\text{为奇数}\\ 1+2^{0.5(r+2)}, & r\text{为偶数但不是4的倍数}\end{cases} \qquad \text{（式 11-35）}$$

当 r 为奇数时，Gold 码族中约有 50%的码序列归一化相关函数值为 $-1/p$。当 r 为偶数但又不是 4 的倍数时，约有 75%的码序列归一化互相关函数值为 $-1/p$。

Gold 码的自相关函数也是三值函数，但是出现的频率不同。另外，同族 Gold 码的互相关函数是三值，而不同族之间的互相关函数是多值函数。

产生 Gold 码可以有两种方法，一种是将对应于优选对的两个移位寄存器串联成 $2r$ 级的线性移位寄存器；另一种方法是将两个移位寄存器并联后模 2 相加。

在优选对产生的 Gold 码末尾加一个 0，使序列长度为偶数，就生成正交 Gold 码（偶位）。

11.4 直接序列扩频通信系统设计

11.4.1 发射机设计

直接序列扩频通信系统的发射机如图 11-7 所示。

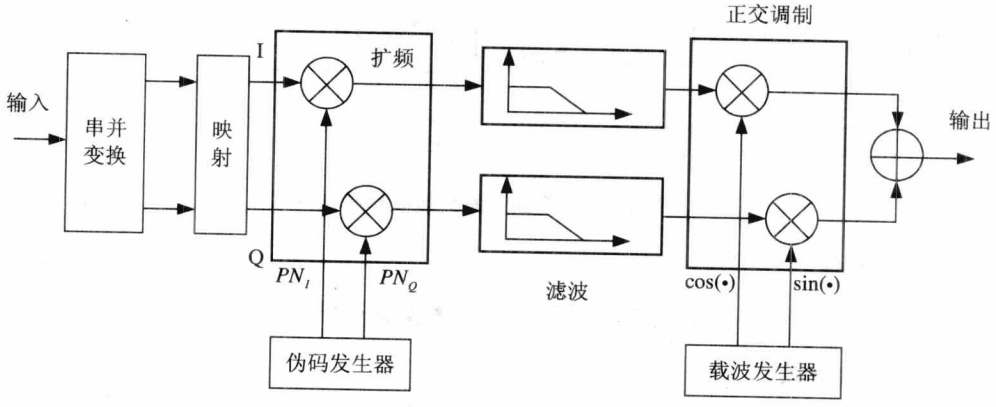


图 11-7　扩频通信系统发射机框图

（1）串并变换

本文采用正交调制方式，所以要进行串并变换分成 I、Q 两路，同时为了消除相位模糊，可以加入差分编码。

（2）映射

差分编码后出来的 I 路和 Q 路数据是由 0 和 1 组成的，需要把 I 路和 Q 路数据联合映射到星座图上的点。

（3）扩频

将 I、Q 两路数据分别与伪码产生器产生的伪码相乘，得到新的数据速率为伪码速率的二进制基带数据，达到扩展频谱的作用。

（4）滤波

数字信号在传输时需要一定的带宽。为了经济地利用频带资源，我们希望信号占用的频带尽可能窄，并且频谱间不应引起码间干扰（ISI），这就需要对数字信号进行频谱成型滤波。

（5）正交调制

I 路和 Q 路信号分别与两个正交的载波信号相乘，将频谱搬移到便于传输的中频段，再将两者相加。

11.4.2 接收机设计

直接序列扩频通信系统的接收机如图 11-8 所示。

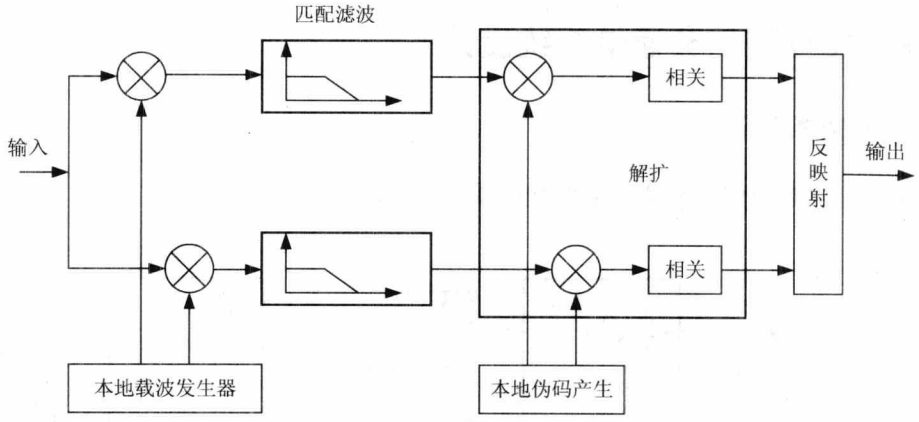


图 11-8 扩频通信系统接收机框图

（1）相干解调

数字信号经过两路正交的载波进行下变频之后重新得到基带信号。

（2）Nyquist 滤波

该滤波器的作用有两点：经过 A/D 变换和下变频的信号含有许多寄生频谱，因而必须用一个低通滤波器予以消除；对接收到的信号进行匹配滤波。

（3）解扩

将接收到的信号与本地伪码进行相关运算，以恢复出原始传输数据。

（4）反映射

将解扩后的数据通过符号判决重新对应为星座图上的点，再对应为 0 和 1 表示的二进制数据。

11.4.3 系统仿真参数

根据前面介绍的 DS-CDMA 通信系统的设计，表 11-1 给出仿真系统的主要参数。

表 11-1 DS-CDMA 通信系统仿真参数

码元速率	256ksps
比特速率	512kbps
调制方式	QPSK
扩频码	m 序列/Gold 序列/正交 Gold 序列
扩频码阶数	3
扩频因子	7
信道	瑞利衰落和加性高斯白噪声（AWGN）

11.4.4　系统性能仿真

根据前面的直接序列扩频通信系统的分析设计（此处为了简化，忽略了射频调制解调部分的设计），可以仿真得到直接序列扩频通信系统的性能如图 11-9 所示。

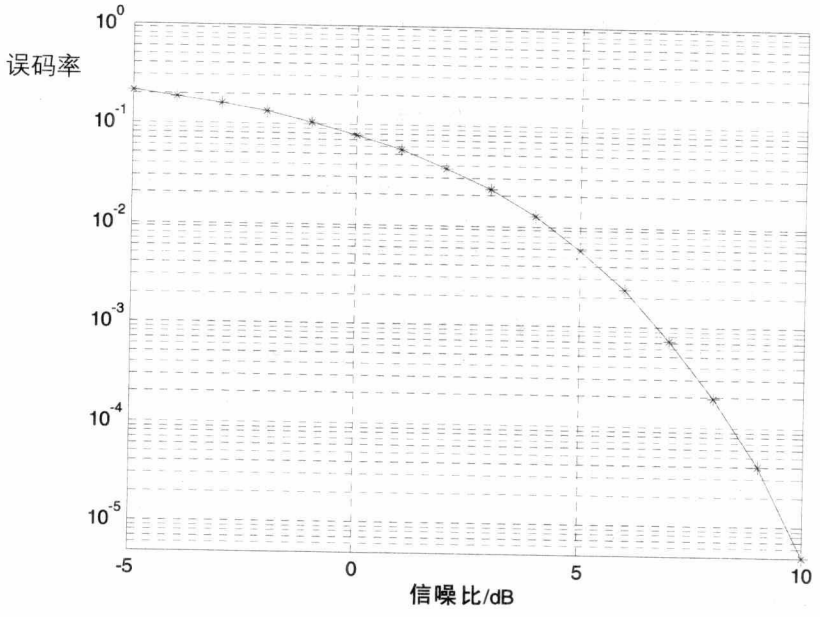


图 11-9　扩频通信系统性能仿真

11.5　直接序列扩频通信系统仿真程序

下面列出直接序列扩频通信系统 DS-CDMA 方式的 MATLAB 仿真程序与代码注释，供读者参考分析。

```
% main_DSCDMA.m
% 这个仿真程序用于实现 DS-CDMA 通信系统仿真
%

%++++++++++++++++++++++++准备部分++++++++++++++++++++++++++

clear all;
clc

sr   = 256000.0;              % 符号速率
ml   = 2;                     % 调制阶数
br   = sr * ml;               % 比特速率
nd   = 100;                   % 符号数
SNR=-5:1:10;                  % Eb/No
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
%+++++++++++++++++++++滤波器初值设定++++++++++++++++++++++

irfn   = 21;                % 滤波器阶数
IPOINT =  8;                % 过采样倍数
alfs   =  0.5;              % 滚降因子
[xh]   = hrollfcoef(irfn,IPOINT,sr,alfs,1);
[xh2]  = hrollfcoef(irfn,IPOINT,sr,alfs,0);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++扩频码初值设定++++++++++++++++++++
user  = 1;                  % 用户数
seq   = 1;                  % 1:m序列  2:Gold序列  3:正交Gold序列
stage = 3;                  % 序列阶数
ptap1 = [1 3];              % 第一个线性移位寄存器的系数
ptap2 = [2 3];              % 第二个线性移位寄存器的系数
regi1 = [1 1 1];            % 第一个线性移位寄存器的初始化
regi2 = [1 1 1];            % 第二个线性移位寄存器的初始化
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

disp('--------------start-------------------');

%+++++++++++++++++扩频码的产生+++++++++++++++++
for ebn0=-5:1:10
switch seq
case 1                                  % m序列
   code = mseq(stage,ptap1,regi1,user);
case 2                                  % Gold序列
   m1   = mseq(stage,ptap1,regi1);
   m2   = mseq(stage,ptap2,regi2);
   code = goldseq(m1,m2,user);
case 3                                  % 正交Gold序列
   m1   = mseq(stage,ptap1,regi1);
   m2   = mseq(stage,ptap2,regi2);
   code = [goldseq(m1,m2,user),zeros(user,1)];
end
code = code * 2 - 1;
clen = length(code);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++信道衰减初值设定++++++++++++++++++++
rfade  = 0;                             % 瑞利衰减 0:不考虑 1:考虑
itau   = [0,8];                         % 延时
dlvl1  = [0.0,40.0];                    % 衰减电平
n0     = [6,7];                         % 用于产生衰落的波数
th1    = [0.0,0.0];                     % 延迟波形的初始相位
itnd1  = [3001,4004];                   % 设置衰落计数
now1   = 2;                             % 主径和延迟波形总数
tstp   = 1 / sr / IPOINT / clen;        % 时间分辨率
fd     = 160;                           % 多普勒频移[Hz]
```

```
flat   = 1;                                  % 平坦瑞利衰落环境
itndel = nd * IPOINT * clen * 30;
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++仿真运算开始++++++++++++++++++++++
nloop = 1000;                                % 仿真循环次数
noe   = 0;
nod   = 0;

for ii=1:nloop
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++发射机++++++++++++++++++++++++++
    data = rand(user,nd*ml) > 0.5;

    [ich, qch]  = qpskmod(data,user,nd,ml);          % QPSK 调制
    [ich1,qch1] = spread(ich,qch,code);              % 扩频
    [ich2,qch2] = compoversamp2(ich1,qch1,IPOINT);   % 过采样
    [ich3,qch3] = compconv2(ich2,qch2,xh);           % 滤波

    if user == 1
       ich4 = ich3;
       qch4 = qch3;
    else
       ich4 = sum(ich3);
       qch4 = sum(qch3);
    end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++衰减信道++++++++++++++++++++++++
    if rfade == 0
       ich5 = ich4;
       qch5 = qch4;
    else
       [ich5,qch5] = sefade(ich4,qch4,itau,dlvl1,th1,n0,itnd1, …
                        now1,length(ich4),tstp,fd,flat);
       itnd1 = itnd1 + itndel;
    end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++接收机++++++++++++++++++++++++++
    spow = sum(rot90(ich3.^2 + qch3.^2)) / nd;    % 衰减计算
    attn = sqrt(0.5 * spow * sr / br * 10^(-ebn0/10));

    [ich6,qch6] = comb2(ich5,qch5,attn);          % 添加高斯白噪声(AWGN)
    [ich7,qch7] = compconv2(ich6,qch6,xh2);       % 滤波

    sampl = irfn * IPOINT + 1;
    ich8  = ich7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
    qch8  = qch7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
```

```
    [ich9 qch9] = despread(ich8,qch8,code);                    % 解扩

    demodata = qpskdemod(ich9,qch9,user,nd,ml);                % QPSK 解调
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++误码率分析++++++++++++++++++++

    noe2 = sum(sum(abs(data-demodata)));
    nod2 = user * nd * ml;
    noe  = noe + noe2;
    nod  = nod + nod2;

%     fprintf('%d\t%e\n',ii,noe2/nod2);

end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++数据文件++++++++++++++++++++++++++
ber = noe / nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
fid = fopen('BER.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
fclose(fid);
err_rate_final(ebn0+6)=ber;
end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++性能仿真图++++++++++++++++++++++++++
figure
semilogy(SNR,err_rate_final,'b-*');
xlabel('信噪比/dB')
ylabel('误码率')
axis([-5,10,0,1])
grid on

disp('--------------end-------------------');
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


% ************************beginning of file*****************************
% hrollfcoef.m
%
% 此函数用于产生 Nyquist 滤波器的系数
%


function [xh] = hrollfcoef(irfn,ipoint,sr,alfs,ncc)
```

```
%+++++++++++++++++++++++variables+++++++++++++++++++++++++++++
% irfn    用于滤波的符号数
% ipoint  每个符号的抽样值数
% sr      符号速率
% alfs    衰减截止频率
% ncc     1 -- 发送端滤波  0 -- 接收端滤波
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

xi=zeros(1,irfn*ipoint+1);
xq=zeros(1,irfn*ipoint+1);

point = ipoint;
tr = sr ;
tstp = 1.0 ./ tr ./ ipoint;
n = ipoint .* irfn;
mid = ( n ./ 2 ) + 1;
sub1 = 4.0 .* alfs .* tr;

for i = 1 : n

  icon = i - mid;
  ym = icon;

  if icon == 0.0
    xt = (1.0-alfs+4.0.*alfs./pi).* tr;
  else
    sub2 =16.0.*alfs.*alfs.*ym.*ym./ipoint./ipoint;
    if sub2 ~= 1.0
      x1=sin(pi*(1.0-alfs)/ipoint*ym)./pi./(1.0-sub2)./ym./tstp;
      x2=cos(pi*(1.0+alfs)/ipoint*ym)./pi.*sub1./(1.0-sub2);
      xt = x1 + x2;
    else
      xt = alfs.*tr.*((1.0-2.0/pi).*cos(pi/4.0/alfs)+(1.0+2.0./pi). *sin
(pi/4.0/alfs))./sqrt(2.0);
    end
  end

  if ncc == 0                          %当接收机情况
    xh( i ) = xt ./ ipoint ./ tr;
  elseif ncc == 1                      %当发射机情况
    xh( i ) = xt ./ tr;
  else
    error('ncc error');
  end

end

%************************end of file*********************************
```

```
% ************************beginning of file*****************************
% mseq.m
%
% 此函数产生 m 序列
%
% 试举一例:
%    stg     = 3
%    taps    = [ 1 , 3 ]
%    inidata = [ 1 , 1 , 1 ]
%    n       = 2
%


function [mout] = mseq(stg, taps, inidata, n)

%+++++++++++++++++++++++variables+++++++++++++++++++++++++++
% stg       m 序列阶数
% taps      线性移位寄存器的系数
% inidata   序列的初始化
% n         输出序列的数目
% mout      输出的 m 序列
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if nargin < 4
   n = 1;
end

mout = zeros(n,2^stg-1);
fpos = zeros(stg,1);

fpos(taps) = 1;

for ii=1:2^stg-1

   mout(1,ii) = inidata(stg);          % 输出数据的存储
   num        = mod(inidata*fpos,2);   % 反馈数据的计算

   inidata(2:stg) = inidata(1:stg-1);  % 线形移位寄存器的一次移位
   inidata(1)     = num;               % 返回反馈值

end

if n > 1
   for ii=2:n
      mout(ii,:) = shift(mout(ii-1,:),1,0);
   end
end

%************************end of file*********************************
```

```
% ************************beginning of file*****************************
% shift.m
%
% 此函数用于实现线性移位寄存器的移位操作
%

function [outregi] = shift(inregi,shiftr,shiftu)

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% inrege    向量或矩阵
% shiftr    右移量
% shiftu    顶部移位量
% outregi   寄存器的输出
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[h, v]  = size(inregi);
outregi = inregi;

shiftr = rem(shiftr,v);
shiftu = rem(shiftu,h);

if shiftr > 0
   outregi(:,1      :shiftr) = inregi(:,v-shiftr+1:v      );
   outregi(:,1+shiftr:v     ) = inregi(:,1        :v-shiftr);
elseif shiftr < 0
   outregi(:,1        :v+shiftr) = inregi(:,1-shiftr:v     );
   outregi(:,v+shiftr+1:v      ) = inregi(:,1      :-shiftr);
end

inregi = outregi;

if shiftu > 0
   outregi(1        :h-shiftu,:) = inregi(1+shiftu:h,     :);
   outregi(h-shiftu+1:h,      :) = inregi(1      :shiftu,:);
elseif shiftu < 0
   outregi(1      :-shiftu,:) = inregi(h+shiftu+1:h,      :);
   outregi(1-shiftu:h,     :) = inregi(1        :h+shiftu,:);
end

%************************end of file*********************************


% ************************beginning of file*****************************
% goldseq.m
%
%此函数用于产生 gold 序列
%
```

```
function [gout] = goldseq(m1, m2, n)

%++++++++++++++++++++++++variables++++++++++++++++++++++++++++
% m1          第一个 m 序列
% m2          第二个 m 序列
% n           输出序列的数目
% gout        输出产生的 gold 序列
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if nargin < 3
   n = 1;
end

gout = zeros(n,length(m1));

for ii=1:n
   gout(ii,:) = xor(m1,m2);
   m2         = shift(m2,1,0);
end

%************************end of file**********************************


% ************************beginning of file*****************************
% qpskmod.m
%
% 此函数实现 QPSK 调制
%


function [iout,qout]=qpskmod(paradata,para,nd,ml)

%++++++++++++++++++++++variables++++++++++++++++++++++++++++
% paradata      输入数据
% iout          输出的实部数据
% qout          输出的虚部数据
% para          并行信道数
% nd            输入数据个数
% ml            调制阶数
% %++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

m2=ml./2;

paradata2=paradata.*2-1;
count2=0;

for jj=1:nd

    isi = zeros(para,1);
    isq = zeros(para,1);
```

```
    for ii = 1 : m2
        isi = isi + 2.^( m2 - ii ) .* paradata2((1:para),ii+count2);
        isq = isq + 2.^( m2 - ii ) .* paradata2((1:para),m2+ii+count2);
    end

    iout((1:para),jj)=isi;
    qout((1:para),jj)=isq;

    count2=count2+m1;

end

%************************end of file*********************************


% ************************beginning of file*****************************
% spread.m
%
%此函数用于实现扩频调制
%


function [iout, qout] = spread(idata, qdata, code1)

%++++++++++++++++++++++variables++++++++++++++++++++++++++++
% idata     输入序列实部
% qdata     输入序列虚部
% iout      输出序列实部
% qout      输出序列虚部
% code1     扩频码序列
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

switch nargin
case { 0 , 1 }
   error('lack of input argument');
case 2
   code1 = qdata;
   qdata = idata;
end

[hn,vn] = size(idata);
[hc,vc] = size(code1);

if hn > hc
   error('lack of spread code sequences');
end

iout = zeros(hn,vn*vc);
qout = zeros(hn,vn*vc);
```

```
for ii=1:hn
    iout(ii,:) = reshape(rot90(code1(ii,:),3)*idata(ii,:),1,vn*vc);
    qout(ii,:) = reshape(rot90(code1(ii,:),3)*qdata(ii,:),1,vn*vc);
end

%************************end of file*********************************


% ************************beginning of file*****************************
% compoversamp2.m
%
% 此函数实现"sample"倍升采样
%

function [iout,qout] = compoversamp2(iin, qin, sample)

%++++++++++++++++++++++++variables++++++++++++++++++++++++++++
% iin       输入序列实部
% qin       输入序列虚部
% iout      输出序列实部
% qout      输出序列虚部
% sample    升采样的倍数
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[h,v] = size(iin);

iout = zeros(h,v*sample);
qout = zeros(h,v*sample);

iout(:,1:sample:1+sample*(v-1)) = iin;
qout(:,1:sample:1+sample*(v-1)) = qin;

%************************end of file*********************************


% ************************beginning of file*****************************
% compconv2.m
%
% 此函数用于实现有用信号的滤波
%


function [iout, qout] = compconv2(idata, qdata, filter)

%++++++++++++++++++++++++variables++++++++++++++++++++++++++++
% idata     输入序列实部
% qdata     输入序列虚部
% iout      输出序列实部
% qout      输出序列虚部
```

```
% filter    滤波器的系数
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
iout = conv2(idata,filter);
qout = conv2(qdata,filter);

%************************end of file*********************************


% ************************beginning of file*****************************
% sefade.m
%
% 此函数用于实现信道的频率选择性衰落设计
%


function[iout,qout,ramp,rcos,rsin]=sefade(idata,qdata,itau,dlvl,th,n0,i
tn,n1,nsamp,tstp,fd,flat)

%+++++++++++++++++++++++++variables++++++++++++++++++++++++++++
% idata     输入的实部数据
% qdata     输入的虚部数据
% iout      输出的实部数据
% qout      输出的虚部数据
% ramp      幅度衰减
% rcos      正交分量衰减
% rsin      同相分量衰减
% itau      各径时延
% dlvl      各多径衰减量
% th        各多径初始相位
% n0        用于产生各径衰落的波数
% itn       各多径衰减计数
% n1        主径和各延迟波形总数
% nsamp     符号数
% tstp      最小时间分辨率
% fd        最大多普勒频移
% flat      是否是平坦衰落
% (1->flat (只有幅度衰落),0->nomal(相位和幅度都衰落)
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

iout = zeros(1,nsamp);
qout = zeros(1,nsamp);

total_attn = sum(10 .^( -1.0 .* dlvl ./ 10.0));

for k = 1 : n1

    atts = 10.^( -0.05 .* dlvl(k));

    if dlvl(k) >= 40.0
```

```
            atts = 0.0;
        end

        theta = th(k) .* pi ./ 180.0;

        [itmp,qtmp] = delay ( idata , qdata , nsamp , itau(k));
        [itmp3,qtmp3,ramp,rcos,rsin] = fade (itmp,qtmp,nsamp, tstp,fd,
n0(k),itn(k),flat);

      iout = iout + atts .* itmp3 ./ sqrt(total_attn);
      qout = qout + atts .* qtmp3 ./ sqrt(total_attn);

    end

    %************************end of file*********************************


    % ************************beginning of file*****************************
    % delay.m
    % 此函数用以实现信号的延迟传输
    %

    function [iout,qout] = delay( idata, qdata , nsamp , idel )

    %+++++++++++++++++++++++variables++++++++++++++++++++++++++++
    % idata     输入序列实部
    % qdata     输入序列虚部
    % iout      输出序列实部
    % qout      输出序列虚部
    % nsamp     仿真的抽样值数量
    % idel      延迟的抽样值数量
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    iout=zeros(1,nsamp);
    qout=zeros(1,nsamp);

    if idel ~= 0
      iout(1:idel) = zeros(1,idel);
      qout(1:idel) = zeros(1,idel);
    end

    iout(idel+1:nsamp) = idata(1:nsamp-idel);
    qout(idel+1:nsamp) = qdata(1:nsamp-idel);

    %************************end of file*********************************


    % ************************beginning of file*****************************
    % fade.m
    %
```

```
    % 此函数实现信道的瑞利衰减设计
    %

    function [iout,qout,ramp,rcos,rsin]=fade(idata,qdata,nsamp,tstp,fd,no,
counter,flat)

    %++++++++++++++++++++++variables++++++++++++++++++++++++++
    % idata      输入序列实部
    % qdata      输入序列虚部
    % iout       输出序列实部
    % qout       输出序列虚部
    % ramp       幅度衰减
    % rcos       正交分量衰减
    % rsin       同相分量衰减
    % nsamp      符号数
    % tstp       最小时间分辨率
    % fd         最大多普勒频移
    % no         用于产生衰落的波数
    % counter    衰落计数
    % flat       是否是平坦衰落
    % (1->flat (只有幅度衰落),0->nomal(相位和幅度都衰落)
    %++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   if fd ~= 0.0
      ac0 = sqrt(1.0 ./ (2.0.*(no + 1)));
      as0 = sqrt(1.0 ./ (2.0.*no));
      ic0 = counter;

      pai = 3.14159265;
      wm = 2.0.*pai.*fd;
      n = 4.*no + 2;
      ts = tstp;
      wmts = wm.*ts;
      paino = pai./no;

      xc=zeros(1,nsamp);
      xs=zeros(1,nsamp);
      ic=[1:nsamp]+ic0;

    for nn = 1: no
        cwn = cos( cos(2.0.*pai.*nn./n).*ic.*wmts );
        xc = xc + cos(paino.*nn).*cwn;
        xs = xs + sin(paino.*nn).*cwn;
    end

    cwmt = sqrt(2.0).*cos(ic.*wmts);
    xc = (2.0.*xc + cwmt).*ac0;
    xs = 2.0.*xs.*as0;

    ramp=sqrt(xc.^2+xs.^2);
```

```
  rcos=xc./ramp;
  rsin=xs./ramp;

  if flat ==1
    iout = sqrt(xc.^2+xs.^2).*idata(1:nsamp);
    qout = sqrt(xc.^2+xs.^2).*qdata(1:nsamp);
  else
    iout = xc.*idata(1:nsamp) - xs.*qdata(1:nsamp);
    qout = xs.*idata(1:nsamp) + xc.*qdata(1:nsamp);
  end

else
  iout=idata;
  qout=qdata;
end

%************************end of file*********************************


% ************************beginning of file*****************************
% comb2.m
%
% 此函数实现信道的高斯白噪声
%

function [iout, qout] = comb2(idata, qdata, attn)

%%++++++++++++++++++++++variables++++++++++++++++++++++++++
% idata    输入序列实部
% qdata    输入序列虚部
% iout     输出序列实部
% qout     输出序列虚部
% attn     根据信噪比得到的信号衰减水平
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

v = length(idata);
h = length(attn);

iout = zeros(h,v);
qout = zeros(h,v);

for ii=1:h
    iout(ii,:) = idata + randn(1,v) * attn(ii);
    qout(ii,:) = qdata + randn(1,v) * attn(ii);
end
%************************end of file*********************************


% ************************beginning of file*****************************
% despread.m
```

```
%
% 此函数实现数据的解频
%


function [iout, qout] = despread(idata, qdata, code1)

%+++++++++++++++++++++++variables+++++++++++++++++++++++++++
% idata     输入序列实部
% qdata     输入序列虚部
% iout      输出序列实部
% qout      输出序列虚部
% code1     扩频码序列
% %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

switch nargin
case { 0 , 1 }
   error('lack of input argument');
case 2
   code1 = qdata;
   qdata = idata;
end

[hn,vn] = size(idata);
[hc,vc] = size(code1);

vn      = fix(vn/vc);

iout    = zeros(hc,vn);
qout    = zeros(hc,vn);

for ii=1:hc
   iout(ii,:) =
rot90(flipud(rot90(reshape(idata(ii,:),vc,vn)))*rot90(code1(ii,:),3));
   qout(ii,:) =
rot90(flipud(rot90(reshape(qdata(ii,:),vc,vn)))*rot90(code1(ii,:),3));
end

%*************************end of file*********************************


% ************************beginning of file*****************************
% qpskdemod.m
%
% 此函数实现 QPSK 解调
%


function [demodata]=qpskdemod(idata,qdata,para,nd,ml)
```

```
%++++++++++++++++++++++++variables++++++++++++++++++++++++++++
% idata     输入数据的实部
% qdata     输入数据的虚部
% demodata  解调后的数据
% para      并行的信道数
% nd        输入数据个数
% ml        调制阶数
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

demodata=zeros(para,ml*nd);
demodata((1:para),(1:ml:ml*nd-1))=idata((1:para),(1:nd))>=0;
demodata((1:para),(2:ml:ml*nd))=qdata((1:para),(1:nd))>=0;

%************************end of file**********************************


% ************************beginning of file*****************************
% autocorr.m
%
% 此函数实现一个序列的自相关运算
%

function [out] = autocorr(indata, tn)

%%++++++++++++++++++++++variables++++++++++++++++++++++++++++
% indata    输入序列
% tn        序列的周期长度
% out       自相关函数
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
if nargin < 2
   tn = 1;
end

ln  = length(indata);
out = zeros(1,ln*tn);

for ii=0:ln*tn-1
   out(ii+1) = sum(indata.*shift(indata,ii,0));
end

%************************end of file**********************************


% ************************beginning of file*****************************
% crosscorr.m
%
% 此函数实现两个序列的互相关运算
%

function [out] = crosscorr(indata1, indata2, tn)
```

```
%++++++++++++++++++++++variables++++++++++++++++++++++++++++
% indata1        第一个输入序列
% indata2        第二个输入序列
% tn             序列的周期长度
% out            互相关运算的结果
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if nargin < 3
   tn = 1;
end

ln  = length(indata1);
out = zeros(1,ln*tn);

for ii=0:ln*tn-1
   out(ii+1) = sum(indata1.*shift(indata2,ii,0));
end

%************************end of file*********************************
```

11.6 本章小结

本章首先对直接序列扩频通信系统原理、直接序列扩频通信系统组成结构进行了介绍，然后以直接序列扩频通信系统的 DS-CDMA 方式为例，分析了具体的 MATLAB 仿真设计过程。读者学习的时候，需要了解直接序列扩频通信系统的特点，重点掌握系统仿真原理的运用，同时吃透仿真程序设计的函数运算，这也有助于对其他类似扩频通信系统的举一反三。

第 12 章　IS-95 前向链路通信系统仿真设计

IS 的全称为 Interim Standard，即暂时标准。IS-95 是由高通公司（Qualcomm）发起的第一个基于 CDMA 的数字蜂窝标准。IS-95 也叫做 TIA-EIA-95。它是一个使用 CDMA 的 2G 移动通信标准，通过数据无线电的多接入方案，进行发送声音、数据和在无线电话和蜂窝站点间发信号数据（如被拨的电话号码）。

IS-95 是最主要的基于 CDMA 技术的 2G 移动通信的空中接口标准分配的编号，IS-95 及其相关标准是最早商用的基于 CDMA 技术的移动通信标准，它与后继的 CDMA2000 经常被简称为 CDMA。本章对 IS-95 前向链路通信系统的仿真原理与设计过程进行详细分析。

12.1 IS-95 系统参数与特性

1993 年 7 月，美国公布了由 Qualcomm 公司提出并获 TIA/EIA 通过的 IS-95 标准。IS-95 标准的全称是“双模宽带扩频蜂窝系统的移动台-基站兼容标准”，实际上这个标准是一个公共空中接口（CAI）——它没有完全规定一个系统怎样实现，而只是提出了信令协议和数据结构的特点与限制。

12.1.1 IS-95 系统参数

表 12-1 列出了 IS-95 空中接口的主要技术参数。

表 12-1　IS-95 系统的主要技术参数

频段	824 ~ 849MHz（上行）；869 ~ 894MHz（下行）
载波间隔	1.25MHz
双工方式	FDD
多址技术	CDMA
帧长度	20ms
数据速率	1200/2400/4800/9600bps
码片速率	1.2288Mcps
信道编码	卷积码，r=1/3，k=9（上行）；r=1/2，k=9（下行）；
调制方式	OQPSK（上行）；QPSK（下行）

12.1.2 IS-95 系统特性

本节将概括介绍 IS-95 系统的一些操作特性，包括频率复用、功率控制、软切换和分

集技术。

有限频率资源的重复利用，是移动通信蜂窝结构的重要特征，也是提高频谱利用率的有效手段。CDMA 使用不同的扩频编码来区分用户，所有用户仅用一个频率进行收发工作。这样，移动通信系统的所有蜂窝都重复使用同一频率，频率的重复利用率为 1。通过频率复用，增加了系统容量。

在 CDMA 数字蜂窝移动通信系统中，由于信道地址码的互相关作用，将产生两方面的影响：一是任何一个信道将受到其他不同地址码信道的干扰，即多址干扰；二是距离接收机近的信道将严重干扰距离接收机远的信道的接收，使近端强信号掩盖了远端弱信号，即远近效应。CDMA 是一种干扰受限系统，各种干扰的存在和积累将直接影响系统的容量和通信质量。因此，基站和移动台的功率必须根据需要时刻变化，使系统既能维持高质量的通信，又能保证对同频段的其他码分信道不产生干扰，这就是功率控制。

当移动用户穿过两个小区间的边界并且必须转换基站时，切换发生。在大多数蜂窝系统中，用户必须在同新基站建立连接前断开同前一基站的连接——即“硬切换”。这是因为两个基站工作在不同的频段上。而在 IS-95 中，所有的小区工作在同一频段。这就意味着，在切换期间移动台可以保持同两个基站的连接——这被称为“软切换”或“断开之前接续”。软切换时，移动台可从两个基站中选择较好的信号。

分集是一种寻找独立的信号路径并合并，以重建发射信号的技术。这种技术以相当低的代价提供对衰落信号的极大改善。IS-95 系统中同时采用了频率、时间和空间分集。

频率分集：扩频特性恰好引入了频率分集。窄带信号通过宽频带扩频。信号解扩后，信道频谱上任何为零的深衰落影响都将减少。

时间分集：数据在发射前通过交织器传送，这在时间上分散了数据比特。任何突发性错误不会影响连续的多个比特。

空间分集：软切换时，移动台根据信号质量在不同的基站间选取。

IS-95 系统中还用到了 RAKE 分集接收技术。RAKE 接收机检测信号的多径时延分量，重新合并以得到更好的接收信号。

12.2 IS-95 前向链路系统设计

在 IS-95 系统中，移动台和基站通过“前向”（基站到移动台）和“反向”（移动台到基站）射频链路通信，也分别称作“下行”和“上行”。前向和反向链路在概念上非常相似，只是在特定的处理阶段细节上有所不同。由于这种相似性，下面的分析与仿真设计只涉及前向链路。

前向链路由四种类型的逻辑信道组成：导频信道为解调其他信道提供相干的相位参考；同步信道将同步信息传送给小区内的所有移动台；寻呼信道用于发送控制信息及通过基站向移动台发送寻呼；业务信道由基站向移动台发送用户信息和信令业务。从信号处理的观点来看，最复杂的是业务信道，其余的都是它的简化形式。

IS-95 中采用变速率语音编码，数据率可以是 1200、2400、4800 和 9600bps。这在系

统中由码元重复来完成。卷积编码之后，数据比特流进入重复编码器。如果原始数据速率低于 9600bps，则重复数据比特使数据速率提高到 9600bps。此处系统设计中没有增加这种复杂度，以固定的 9600bps 速率传输数据。

12.2.1 发射机设计

在 IS-95 CDMA 系统中，信号在信道中是以帧的形式来传送的，帧结构随着信道种类的不同和数据率的不同而变化。

如图 12-1 所示是前向业务信道的帧结构。其中，*F* 表示循环冗余校验帧质量指示器，*T* 表示编码器拖尾比特。传输速率为 9600bps 时，在 20 毫秒的帧持续时间内可以发送 192 个比特。这 192 个比特由 172 个信息比特、12 个帧质量指示比特和 8 个编码拖尾比特组成。帧质量指示比特就是奇偶校验比特，应用于循环冗余编码的系统检错方案中。

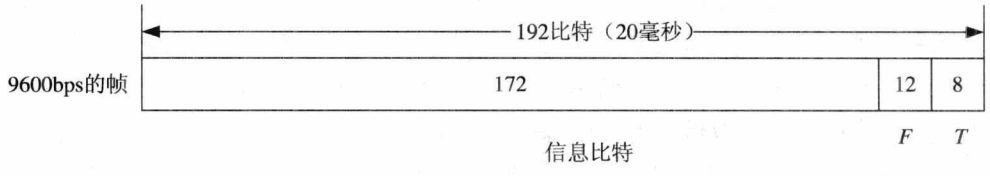


图 12-1 前向业务信道 9600bps 的帧结构

根据 IS-95 前向业务信道结构框图，发射机部分所采用的系统设计框图如图 12-2 所示。

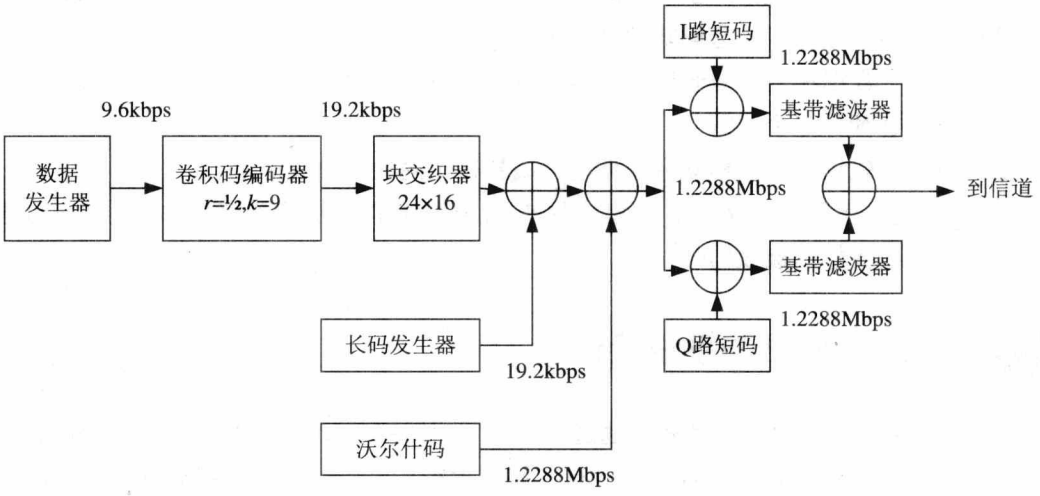


图 12-2 发射机系统框图

数据发生器以每帧 192 比特产生数据作为系统的信源。其中，前 184 位是随机产生的二进制信息比特，最后 8 位全零的编码器尾比特用于实现卷积编码。经码率为 1/2、约束长度为 9 的卷积编码器后，每帧的调制码元数为 192×2=384。对于 9.6kbps 的数据率，不需要进行码元重复。块交织器将输入的数据存为 16 列×24 行，一次处理一帧内的 384 个符号。为了提供保密性，交织后的数据与长码发生器产生的长码模 2 加，以对数据进行加扰。

长码的抽样及功率控制予以简化。两两正交的 64 个沃尔什序列中的一个再与数据流进行模 2 加，以提供对同一射频载波的复用。两路短码作为 I/Q 正交序列对数据进行四相扩频，再经过基带滤波后，I、Q 两路信号合成待发送的信号送至信道。在系统仿真中，省略了 QPSK 的射频调制部分。

1. 卷积编码

卷积码是将发送的信息序列通过一个线性的、有限状态的移位寄存器产生的。通常，该移位寄存器由 k 级（每级 k 比特）和 n 个线性的代数函数构成。二进制数据移位输入到编码器，沿着移位寄存器每次移动 k 比特位。每一个 k 比特长的输入序列对应一个 n 比特长的输出序列。因此其编码效率定义为 $R_c = k / n$。参数 k 称为卷积码的约束长度。

从 IS-95 前向链路业务信道图中可以看出，前向链路使用的卷积码编码率为 1/2，约束长度 k=9。IS-95 规定了产生这种码的编码器，如图 12-3 所示。这种码的生成函数为：

$$g_0 = (111101011) = (753)_o$$
$$g_1 = (101110001) = (561)_o \qquad \text{（式 12-1）}$$

对输入到编码器的每一数据比特，生成两个码符号。这些码符号应这样输出，即由生成函数 g_0 编码的码符号 c_0 先输出，由生成函数 g_1 编码的码符号 c_1 后输出。初始化时，卷积编码器应该是全零状态。初始化后的第一个码符号应该由生成函数 g_0 编码。

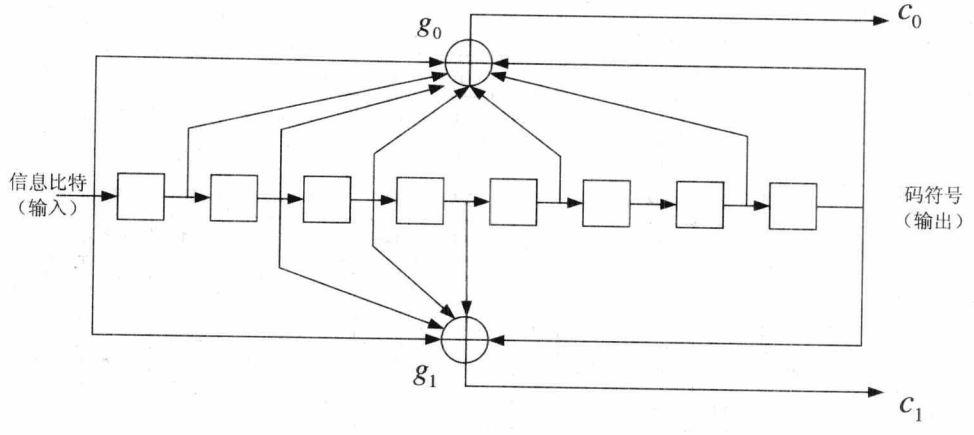


图 12-3 前向链路的卷积编码器：r=1/2，k=9

2. 块交织

交织常与编码或重复相结合，是一种防止突发错误的时间分集形式。符号在进入突发信道之前被改变顺序或进行交织。如果传送时发生突发错误，恢复原序就可以在时间上分散错误。如果交织器设计良好，那么错误将会随机分布，用编码技术就更容易纠正。

最常用的交织技术有两类。最常见的类型是块交织。这种方式常在数据分块分帧的情况下使用，如 IS-95 系统。另一种卷积交织是对连续数据流来说比较实用的类型。块交织很容易实现，而卷积交织有很好的性能。

一个（I,J）的块交织器可以看成是一个 J 行 I 列的存储矩阵。数据按列写入，按行读

出。符号从矩阵的左上角开始写入，从右下角开始读出。连续的数据处理要求有两个矩阵：一个用于数据写入，另一个用于数据读出。解交织过程也要求有两个矩阵，用于反转交织过程。

IS-95 规定，9.6kbps 的前向业务信道在一个 20 毫秒 384 个符号的帧上用表 12-2 中所示的（16,24）输出矩阵，输入矩阵按列顺序写入作交织器。

表 12-2　9.6kbps 前向业务信道的交织器输出序列

1	9	5	13	3	11	7	15	2	10	6	14	4	12	8	16
65	73	69	77	67	75	71	79	66	74	70	78	68	76	72	80
129	137	133	141	131	139	135	143	130	138	134	142	132	140	136	144
193	201	197	205	195	203	199	207	194	202	198	206	196	204	200	208
257	265	261	269	259	267	263	271	258	266	262	270	260	268	264	272
321	329	325	333	323	331	327	335	322	330	326	334	324	332	328	336
33	41	37	45	35	43	39	47	34	42	38	46	36	44	40	48
97	105	101	109	99	107	103	111	98	106	102	110	100	108	104	112
161	169	165	173	163	171	167	175	162	170	166	174	164	172	168	176
225	233	229	237	227	235	231	239	226	234	230	238	228	236	232	240
289	297	293	301	291	299	295	303	290	298	294	302	292	300	296	303
353	361	357	365	355	363	359	367	354	362	358	366	356	364	360	368
17	25	21	29	19	27	23	31	18	26	22	30	20	28	24	32
81	89	85	93	83	91	87	95	82	90	86	94	84	92	88	96
145	153	149	157	147	155	151	159	146	154	150	158	148	156	152	160
209	217	213	221	211	219	215	223	210	218	214	222	212	220	216	224
273	281	277	285	275	283	279	287	274	282	278	286	276	284	280	288
337	345	341	349	339	347	343	351	338	346	342	350	340	348	344	352
49	57	53	61	51	59	55	63	50	58	54	62	52	60	56	64
113	121	117	125	115	123	119	127	114	122	118	126	116	124	120	128
177	185	181	189	179	187	183	191	178	186	182	190	180	188	184	192
241	249	245	253	243	251	247	255	242	250	246	254	244	252	248	256
305	313	309	317	307	315	311	319	306	314	310	318	308	316	312	320
369	377	373	381	371	379	375	383	370	378	374	382	372	380	376	384

3．数据加扰

无线通信的一个主要问题是任何传输都可被窃听者轻易地获得。为了加强 IS-95 传输的保密性，加扰过程将一串密码加到外发数据上。编码过程由称为长码的密钥来完成。只有知道正确的随机数初始值，接收机才能重建长码并解密消息。

长 PN 码序列的速率为 1.2288Mbps，通过对每 64 个 PN 码片进行一次抽样，速率降低为 19.2kbps。长 PN 码是用 42 阶移位寄存器来产生的，周期是 $2^{42}-1\approx 4.4\times 10^{12}$ 码片（在

1.2288Mcps 的速率下将持续 41 天），其线性递归所依据的特征多项式为：

$$p(x)=x^{42}+x^{35}+x^{33}+x^{31}+x^{27}+x^{26}+x^{25}+x^{22}+x^{21}+x^{19}+x^{18}+x^{17}+x^{16}+x^{10}+x^{7}+x^{6}+x^{5}+x^{3}+x^{2}+x+1 \quad (式\ 12\text{-}2)$$

如图 12-4 所示，长码的每一码片由 42 比特掩码同序列发生器的 42 比特状态矢量进行模 2 加产生。

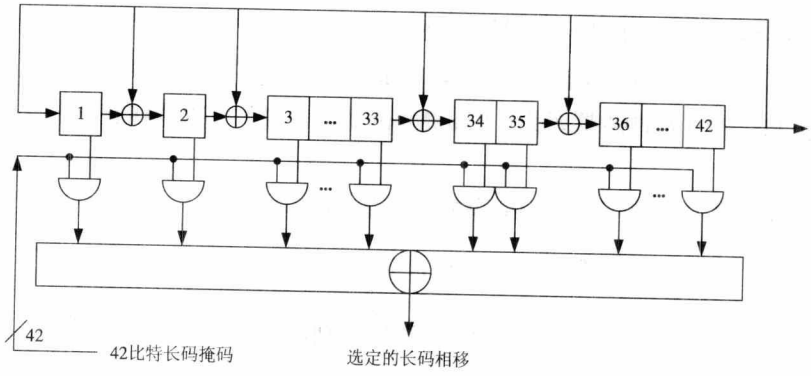


图 12-4 长码发生器

4. 正交复用

在前向链路中，每个信道通过其专用的正交沃尔什序列来区别于其他信道。前向链路的信道由导频信道、同步信道、寻呼信道和业务信道组成。每个信道由信道特定的沃尔什序列调制，沃尔什序列记为 H_i，其中 $i=0,1,\cdots,63$。IS-95 标准将 $\boldsymbol{H}_0$ 分配给导频信道，将 $\boldsymbol{H}_{32}$ 分配给同步信道，$\boldsymbol{H}_1$ 到 $\boldsymbol{H}_7$ 分配给寻呼信道，其余的 $\boldsymbol{H}_i$ 分配给业务信道。

沃尔什序列是维数为 2 的幂的哈达玛矩阵中的某一行，当在一个周期长度上进行相关时它们是正交的。$2N$ 阶哈达玛矩阵可以由递推公式产生：

$$\boldsymbol{H}_1=[1] \quad (式\ 12\text{-}3)$$

$$\boldsymbol{H}_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \quad (式\ 12\text{-}4)$$

$$\boldsymbol{H}_{2N}=\begin{bmatrix}H_N & H_N\\ H_N & \overline{H_N}\end{bmatrix} \quad (式\ 12\text{-}5)$$

这里规定 $\overline{\boldsymbol{H}_N}$ 为 $\boldsymbol{H}_N$ 取负（为其补值）。

在前向业务信道中，19.2kbps 的数据流中的每一输入比特与指定的 64 阶沃尔什序列逐个进行模 2 加，映射为 64 个输出比特。因而，这个过程的输出速率为 1.2288Mbps。

5．正交扩频

IS-95 中使用了两个修正后的短 PN 序列，用于对 QPSK 的同相与正交支路进行扩频。两个短 PN 码是由 15 阶移位寄存器产生的 m 序列，并且每个周期在 PN 序列的特定位置插入一个额外的“0”。因此修正后的短 PN 码周期为 $2^{15}=32768$ 个码片。该序列称为引导 PN 序列，作用是识别不同的基站。不同的基站使用相同的引导 PN 序列，但是各自采用不同的相位偏置。

IS-95 中采用在长为 n-1 的行程后面插入一个 0，这样做有两个目的：一是使不同基站使用的 PN 序列有一部分保持正交；二是使 15 级 PN 序列发生器的周期变为 $2^{15}=32768$ 个码片，这样当 PN 序列的时钟频率为 1.2288Mbps 时，每 2 秒的间隔内序列发生器可以循环 75 次。

同相支路（I 路）所使用的短 PN 码的特征多项式为：

$$P_I(x)=x^{15}+x^{13}+x^9+x^8+x^7+x^5+1 \qquad (式\ 12\text{-}6)$$

正交支路（Q 路）所使用的短 PN 码的特征多项式为：

$$P_Q(x)=x^{15}+x^{12}+x^{11}+x^{10}+x^6+x^5+x^4+x^3+1 \qquad (式\ 12\text{-}7)$$

如图 12-5 所示，使用模板来产生特定相位的短 PN 序列。

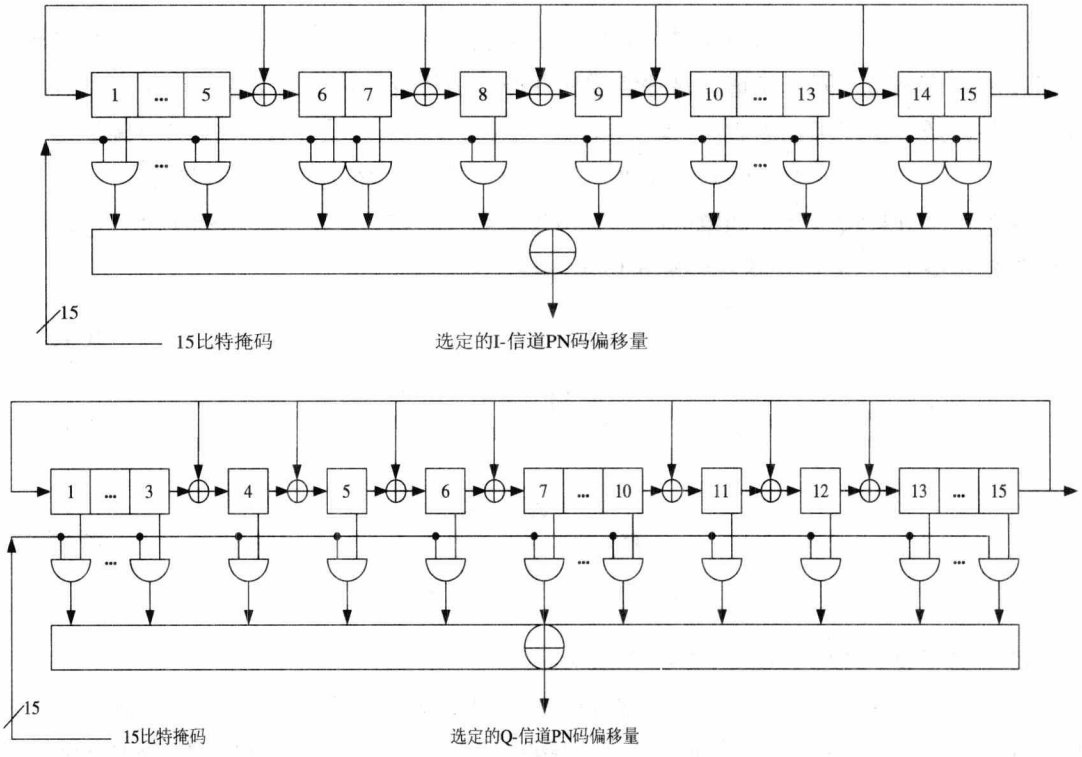


图 12-5　I、Q 支路短码发生器

6. 基带滤波

在现代数字通信系统中，数字化的数据信号必须通过某种适当波形的连续脉冲成型进行发射以完成它在信道内的传播。满足频谱在限定的频带内同时减少或消除 ISI（符号间干扰）是基带波形设计的核心问题。

IS-95 系统中使用的基带成形滤波器满足图 12-6 限制的频率响应 $S(f)$，即通带（$0 \leqslant f \leqslant f_p = 590\text{kHz}$）波纹不大于 1.5dB，阻带（$f > f_s = 740\text{kHz}$）衰减不大于 40dB。除了这些频域的限制，IS-95 还规定滤波器的冲激响应与响应为 $h(k)$的 48 抽头的 FIR 滤波器相近。

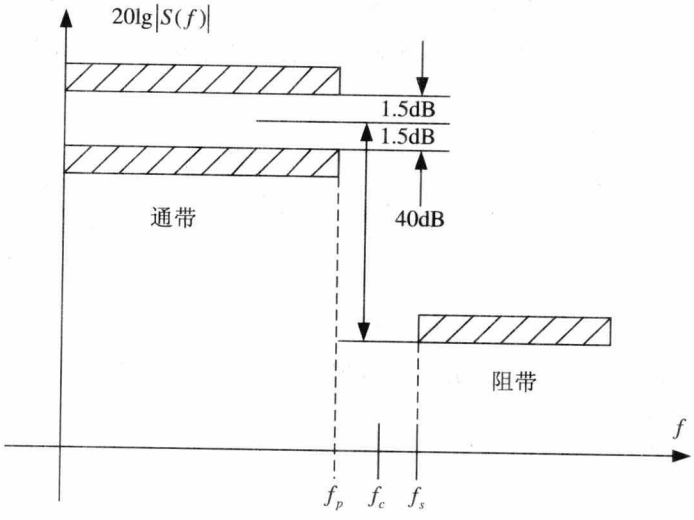


图 12-6　基带滤波器频率响应限制

12.2.2　信道设计

与其他通信信道相比，移动信道是最为复杂的一种。复杂、恶劣的传播条件是移动信道的特征，这是由在运动中进行无线通信这一方式本身决定的。

数字通信信道中用于分析的最简单的模型是加性高斯白噪声信道（AWGN，Additive White Gaussian Noise）。在加性高斯白噪声信道模型中，假定除了高斯白噪声的加入外，不存在失真和其他影响。高斯白噪声是由接收机中的随机电子运动产生的热噪声。在如图 12-7 所示的模型中，发送信号 $s(t)$ 被加性高斯白噪声过程 $n(t)$ 恶化，接收信号 $r(t)$ 表示为：

$$r(t) = s(t) + n(t) \qquad \text{（式 12-8）}$$

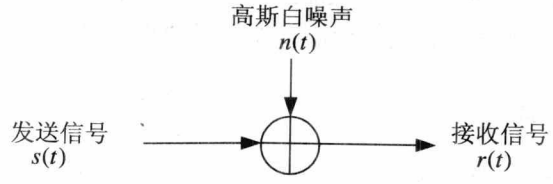


图 12-7　加性高斯白噪声信道

可以使用如下所述的方法产生高斯分布的随机变量作为噪声源。高斯分布的概率密度函数由下式给出：

$$f(C)=\frac{1}{\sqrt{2\pi\sigma}}\mathrm{e}^{-C^2/(2\sigma^2)}\ ,\qquad -\infty<C<\infty \tag{式 12-9}$$

式中 σ^2 是 C 的方差。概率分布函数 $F(C)$ 是在区间 $(-\infty,C)$ 内 $f(C)$ 下所包围的面积，即：

$$F(C)=\int_{-\infty}^{C}f(x)\mathrm{d}x \tag{式 12-10}$$

不幸的是式 12-10 的积分无法用简单函数来表示，这样一来，完成逆映射就很困难。克服这个难题的一种方法已经找到。由概率论知道，具有概率分布函数为

$$F(R)=\begin{cases}0, & R<0\\ 1-e^{-R^2/(2\sigma^2)}, & R\geqslant 0\end{cases} \tag{式 12-11}$$

的瑞利分布的随机变量 R 与一对高斯随机变量 C 和 D 是通过如下变换

$$\begin{aligned}C&=R\cos\theta\\ D&=R\sin\theta\end{aligned} \tag{式 12-12}$$

关联的。这里 θ 是在 $(0,2\pi)$ 内均匀分布的变量，参数 σ^2 是 C 和 D 的方差。因为式 12-11 容易求得逆函数，令

$$F(R)=1-e^{-R^2/(2\sigma^2)}=A \tag{式 12-13}$$

则

$$R=\sqrt{2\sigma^2\ln\left(\frac{1}{1-A}\right)} \tag{式 12-14}$$

式中 A 是在(0,1)内均匀分布的随机变量。现在，如果我们产生了第 2 个均匀分布的随机变量 B，而定义

$$\theta=2\pi B \tag{式 12-15}$$

那么，从式 12-12 可求得两个统计独立的高斯分布随机变量 C 和 D。

12.2.3 接收机设计

接收部分从信道接收信号。经基带滤波、短码解扩、沃尔什解调、解扰、去交织和维特比译码，输出解调后的信号。各通信模块和发射机的相关模块设计类似，这里不再讨论。

12.2.4 系统性能仿真

根据前面介绍的 IS-95 前向链路的系统设计，可以仿真得到系统的性能如图 12-8 所示。

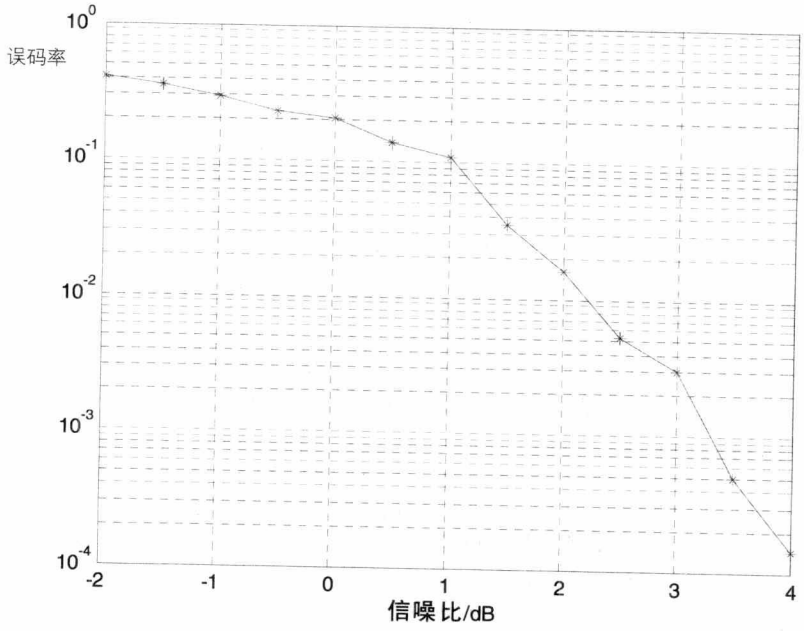


图 12-8 IS-95 前向链路通信系统性能仿真

12.3 IS-95 前向链路系统仿真程序

下面列出 IS-95 前向链路系统的 MATLAB 仿真程序及注释，供读者借鉴参考。

```
%main_IS95_forward.m
%此函数用于 IS-95 前向链路系统的仿真，包括扩
%频调制、匹配滤波、RAKE 接收等相关通信模块。
%仿真环境：加性高斯白噪声信道。
%数据速率 = 9600 kbps
%

clear all
close all
clc

disp('--------------start--------------------');
global Zi Zq Zs show R Gi Gq

clear j;
show = 0;
SD = 0;          % 选择软/硬判决接收

%--------------------主要的仿真参数设置-------------------

BitRate = 9600;
ChipRate = 1228800;
N = 184;
```

```
MFType = 1;  % 匹配滤波器类型—升余弦
R = 5;

%+++++++++++++++++++Viterbi 生成多项式+++++++++++++++++
G_Vit = [1 1 1 1 0 1 0 1 1; 1 0 1 1 1 0 0 0 1];
K = size(G_Vit, 2);
L = size(G_Vit, 1);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++Walsh 矩阵++++++++++++++++++++++

WLen = 64;
Walsh = reshape([1;0]*ones(1, WLen/2), WLen , 1);
%Walsh = zeros(WLen ,1);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++扩频调制 PN 码的生成多项式++++++++++++++
%Gi = [ 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1]';
%Gq = [ 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1]';

Gi_ind = [15, 13, 9, 8, 7, 5, 0]';
Gq_ind = [15, 12, 11, 10, 6, 5, 4, 3, 0]';

Gi = zeros(16, 1);
Gi(16-Gi_ind) = ones(size(Gi_ind));
Zi = [zeros(length(Gi)-1, 1); 1];
% I 路信道 PN 码生成器的初始状态

Gq = zeros(16, 1);
Gq(16-Gq_ind) = ones(size(Gq_ind));
Zq = [zeros(length(Gq)-1, 1); 1];
% Q 路信道 PN 码生成器的初始状态
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++扰码生成多项式+++++++++++++++++++++
Gs_ind = [42, 35, 33, 31, 27, 26, 25, 22, 21, 19, 18, 17, 16, 10, 7, 6, 5,
3, 2, 1, 0]';
Gs = zeros(43, 1);
Gs(43-Gs_ind) = ones(size(Gs_ind));
Zs = [zeros(length(Gs)-1, 1); 1];
% 长序列生成器的初始状态
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++AWGN 信道++++++++++++++++++++++++
EbEc = 10*log10(ChipRate/BitRate);
EbEcVit = 10*log10(L);
EbNo = [-2 : 0.5 : 6.5];          %仿真信噪比范围(dB)
% EbNo = [2 : 0.5 : 2.5];
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
%-------------------------------------------------------

%-------------------------主程序-------------------------

ErrorsB = []; ErrorsC = []; NN = [];
if (SD == 1)
   fprintf('\n SOFT Decision Viterbi Decoder\n\n');
else
   fprintf('\n HARD Decision Viterbi Decoder\n\n');
end

for i=1:length(EbNo)
   fprintf('\nProcessing %1.1f (dB)', EbNo(i));
   iter = 0;ErrB = 0; ErrC = 0;
   while (ErrB <300) & (iter <150)
      drawnow;

      %++++++++++++++++++++++发射机++++++++++++++++++++++
      TxData = (randn(N, 1)>0);
      % 速率为 19.2Kcps
      [TxChips, Scrambler] = PacketBuilder(TxData, G_Vit, Gs);
      % 速率为 1.2288Mcps
      [x PN MF] = Modulator(TxChips, MFType, Walsh);
      %++++++++++++++++++++++++++++++++++++++++++++++++++++

      %++++++++++++++++++++++信道++++++++++++++++++++++++++
      noise = 1/sqrt(2)*sqrt(R/2)*( randn(size(x)) + j*randn
(size(x)))*10^(-(EbNo(i) - EbEc)/20);
      r = x+noise;
      %++++++++++++++++++++++++++++++++++++++++++++++++++++

      %++++++++++++++++++++++接收机++++++++++++++++++++++++
      RxSD = Demodulator(r, PN, MF, Walsh); %软判决，速率为 19.2 Kcps
      RxHD = (RxSD>0);                      % 定义接收码片的硬判决
      if (SD)
        [RxData Metric]= ReceiverSD(RxSD, G_Vit, Scrambler); %软判决
      else
        [RxData Metric]= ReceiverHD(RxHD, G_Vit, Scrambler); %硬判决
      end
     %++++++++++++++++++++++++++++++++++++++++++++++++++++

      if(show)
         subplot(311); plot(RxSD, '-o'); title('Soft Decisions');
         subplot(312); plot(xor(TxChips, RxHD), '-o'); title('Chip Errors');
         subplot(313); plot(xor(TxData, RxData), '-o');
         title(['Data Bit Errors. Metric = ', num2str(Metric)]);
         pause;
      end

      if(mod(iter, 50)==0)
```

```
            fprintf('.');
            save TempResults ErrB ErrC N iter
        end

        ErrB = ErrB + sum(xor(RxData, TxData));
        ErrC = ErrC + sum(xor(RxHD, TxChips));
        iter = iter+ 1;
    end

    ErrorsB = [ErrorsB; ErrB];
    ErrorsC = [ErrorsC; ErrC];
    NN = [NN; N*iter];
    save SimData *
end

%++++++++++++++++++++++++++误码率计算++++++++++++++++++++++++++
PerrB = ErrorsB./NN;
%PerrB1 = ErrorsB1./NN1;
PerrC = ErrorsC./NN;
Pbpsk= 1/2*erfc(sqrt(10.^(EbNo/10)));
PcVit= 1/2*erfc(sqrt(10.^((EbNo-EbEcVit)/10)));
Pc =  1/2*erfc(sqrt(10.^((EbNo-EbEc)/10)));
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%%++++++++++++++++++++++++性能仿真显示++++++++++++++++++++++
figure;
semilogy(EbNo(1:length(PerrB)), PerrB, 'b-*'); hold on;
% %semilogy(EbNo(1:length(PerrB1)), PerrB1, 'k-o'); hold on;
% semilogy(EbNo(1:length(PerrC)), PerrC, 'b-o'); grid on;
% semilogy(EbNo, Pbpsk, 'b-.^');
% %semilogy(EbNo, PcVit, 'k-.x'); ylabel('BER');
% semilogy(EbNo, Pc, 'b-.x');
xlabel('信噪比/dB');
ylabel('误码率');
grid on;
% legend('Pb of System (HD)', 'Pb of System (SD)', 'Pc before Viterbi of
System',
%  ... 'Pb of BPSK with no Viterbi (theory)', 'Pc on Receiver (theory)');
%
%
%
% legend('Pb of System', 'Pc before Viterbi of System', ...
%'Pb of BPSK with no Viterbi (theory)',
%'Pc before Viterbi (theory)', 'Pc on Receiver (theory)');
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

disp('--------------end-------------------');
%-------------------------------------------------------


% ************************beginning of file*****************************
```

```
%PacketBuilder.m
function [ChipsOut, Scrambler] = PacketBuilder(DataBits, G, Gs);
%
%此函数用于产生 IS-95 前向链路系统的发送数据包

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% DataBits          发送数据（二进制形式）
% G                 Viterbi 编码生成多项式
% Gs                长序列生成多项式（扰码生成多项式）
% ChipsOut          输入到调制器的码序列（二进制形式）
% Scrambler         扰码
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

global Zs

K = size(G, 2);
L = size(G, 1);

N = 64*L*(length(DataBits)+K-1);% 码片数 (9.6 Kbps -> 1.288 Mbps)

chips = VitEnc(G, [DataBits; zeros(K-1,1)]);        % Viterbi 编码

% 交织编码
INTERL = reshape(chips, 24, 16);                    % IN: 列, OUT: 行
chips = reshape(INTERL', length(chips), 1);         %速率=19.2 KBps

% 产生扰码
[LongSeq Zs] = PNGen(Gs, Zs, N);
Scrambler = LongSeq(1:64:end);

ChipsOut = xor(chips, Scrambler);
%************************end of file*********************************


% ************************beginning of file*****************************
%VitEnc.m
function y = VitEnc(G, x);

% 此函数根据生成多项式进行 Viterbi 编码
%
%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% G     生成多项式的矩阵
% x     输入数据（二进制形式）
% y     Viterbi 编码输出序列
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

K = size(G, 1);
L = length(x);
```

```
yy = conv2(G, x');
yy = yy(:, 1:L);
y = reshape(yy,K*L, 1);

y = mod(y, 2);
% ************************end of file************************************


% ************************beginning of file*****************************
%PNGen.m
function [y, Z] = PNGen(G, Zin, N);
%
% 此函数是根据生成多项式和输入状态产生长度为 N 的伪随机序列

%++++++++++++++++++++++variables+++++++++++++++++++++++++++
% G           生成多项式
% Zin         移位寄存器初始化
% N           PN 序列长度
% y           生成的 PN 码序列
% Z           移位寄存器的输出状态
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

L = length(G);
Z = Zin;    % 移位寄存器的初始化

y = zeros(N, 1);
for i=1:N
   y(i) = Z(L);
   Z = xor(G*Z(L), Z);
   Z = [Z(L); Z(1:L-1)];
end
%yy = filter(1, flipud(G), [1; zeros(N-1, 1)]);
%yy = mod(yy, 2);
%************************end of file************************************


% ************************beginning of file*****************************
%Modulator.m
function [TxOut, PN, MF] = Modulator(chips, MFType, Walsh);
%
%此函数用于实现 IS-95 前向链路系统的数据调制

%++++++++++++++++++++++variables+++++++++++++++++++++++++++
% chips          发送的初始数据
% MFType         成型滤波器的类型选择
% Walsh          Walsh 码
% TxOut          调制输出信号序列
% PN             用于扩频调制的 PN 码序列
% MF             匹配滤波器参数
```

```
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

global Zi Zq show R Gi Gq

N = length(chips)*length(Walsh);

% 输入速率 = 19.2 kbps, 输出速率= 1.2288 Mcps
tmp = sign(Walsh-1/2)*sign(chips'-1/2);
chips = reshape(tmp, prod(size(tmp)), 1);

[PNi Zi] = PNGen(Gi, Zi, N);
[PNq Zq] = PNGen(Gq, Zq, N);

PN = sign(PNi-1/2) + j*sign(PNq-1/2);

chips_out = chips.*PN;

chips = [chips_out, zeros(N, R-1)];
chips = reshape(chips.' , N*R, 1);

%成型滤波器
switch (MFType)
case 1
   %升余弦滤波器
   L = 25;
   L_2 = floor(L/2);
   n = [-L_2:L_2];
   B = 0.7;
   MF = sinc(n/R).*(cos(pi*B*n/R)./(1-(2*B*n/R).^2));
   MF = MF/sqrt(sum(MF.^2));

case 2
   %矩形滤波器
   L = R;
   L_2 = floor(L/2);
   MF = ones(L, 1);
   MF = MF/sqrt(sum(MF.^2));

case 3
   %汉明滤波器
   L = R;
   L_2 = floor(L/2);
   MF = hamming(L);
   MF = MF/sqrt(sum(MF.^2));

end

MF = MF(:);

TxOut = sqrt(R)*conv(MF, chips)/sqrt(2);
TxOut = TxOut(L_2+1: end - L_2);
```

```
if (show)
   figure;
   subplot(211); plot(MF, '-o'); title('Matched Filter'); grid on;
   subplot(212); psd(TxOut, 1024, 1e3, 113); title('Spectrum');
end
% ************************end of file***********************************


% ************************beginning of file*****************************
%Demodulator.m
function [SD] = Demodulator(RxIn, PN, MF, Walsh);
%
% 此函数是实现基于 RAKE 接收机的 IS-95 前向信链路系统的数据包的解调

%+++++++++++++++++++++++variables+++++++++++++++++++++++++++
% RxIn          输入信号
% PN            PN 码序列（用于解扩）
% MF            匹配滤波器参数
% Walsh         用于解调的 Walsh 码
% SD            RAKE 接收机的软判决输出
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

global R
N = length(RxIn)/R;

L = length(MF);
L_2 = floor(L/2);
rr = conv(flipud(conj(MF)), RxIn);
rr = rr(L_2+1: end - L_2);

Rx = sign(real(rr(1:R:end))) + j*sign(imag(rr(1:R:end)));

Rx = reshape(Rx, 64, N/64);

Walsh = ones(N/64, 1)*sign(Walsh'-1/2);
PN = reshape(PN, 64, N/64)';
PN = PN.*Walsh;

% 输入速率 = 1.2288 Mpbs, 输出速率 = 19.2 kbps
SD= PN*Rx;
SD= real(diag(SD));
% ************************end of file***********************************


% ************************beginning of file*****************************
%ReceiverSD.m
function [DataOut, Metric] = ReceiverSD(SDchips, G, Scrambler);
%
% 此函数用于实现基于 Viterbi 译码的发送数据的恢复
```

```
%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% SDchips              软判决 RAKE 接收机输入符号
% G                    Viterbi 编码生成多项式矩阵
% Scrambler           扰码序列
% DataOut             接收数据（二进制形式）
% Metric              Viterbi 译码最佳度量
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if (nargin == 1)
   G = [1 1 1 1 0 1 0 1 1; 1 0 1 1 1 0 0 0 1];
end

% 速率=19.2 kbps
SDchips = SDchips.*sign(1/2-Scrambler);

INTERL = reshape(SDchips, 16, 24);
SDchips = reshape(INTERL', length(SDchips), 1);  % 速率=19.2 kbps

[DataOut Metric] = SoftVitDec(G, SDchips, 1);
% ************************end of file*********************************


% ************************beginning of file*****************************
%SoftVitDec.m
function [xx, BestMetric] = SoftVitDec(G, y, ZeroTail);

%
% 此函数是实现软判决输入的 Viterbi 译码

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% G              生成多项式的矩阵
% y              输入的待译码序列
% ZeroTail       判断是否包含‘0’尾
% xx             Viterbi 译码输出序列
% BestMetric     最后的最佳度量
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

L = size(G, 1);      % 输出码片数
K= size(G, 2);       % 生成多项式的长度
N = 2^(K-1);         % 状态数
T = length(y)/L;     % 最大栅格深度

OutMtrx = zeros(N, 2*L);
for s = 1:N
    in0 = ones(L, 1)*[0, (dec2bin((s-1), (K-1))-'0')];
    in1 = ones(L, 1)*[1, (dec2bin((s-1), (K-1))-'0')];

    out0 = mod(sum((G.*in0)'), 2);
```

```
        out1 = mod(sum((G.*in1)'), 2);

        OutMtrx(s, :) = [out0, out1];
    end
    OutMtrx = sign(OutMtrx-1/2);

    PathMet = [100; zeros((N-1), 1)];        % 初始状态 = 100
    PathMetTemp = PathMet(:,1);

    Trellis = zeros(N, T);
    Trellis(:,1) = [0 : (N-1)]';

    y = reshape(y, L, length(y)/L);
    for t = 1:T

        yy = y(:, t);
        for s = 0:N/2-1
            [B0 ind0] = max(  PathMet(1+[2*s, 2*s+1]) + [OutMtrx(1+2*s, 0+[1:L])
* yy; OutMtrx(1+(2*s+1), 0+[1:L])*yy] );
            [B1 ind1] = max(  PathMet(1+[2*s, 2*s+1]) + [OutMtrx(1+2*s, L+[1:L])
* yy; OutMtrx(1+(2*s+1), L+[1:L]) * yy] );

            PathMetTemp(1+[s, s+N/2]) =  [B0; B1];
            Trellis(1+[s, s+N/2], t+1) = [2*s+(ind0-1); 2*s + (ind1-1)];
        end
       PathMet = PathMetTemp;

    end

    xx = zeros(T, 1);
    if (ZeroTail)
        BestInd = 1;
    else
        [Mycop, BestInd]  = max(PathMet);
    end

    BestMetric = PathMet(BestInd);
    xx(T) = floor((BestInd-1)/(N/2));

    NextState = Trellis(BestInd, (T+1));
    for t=T:-1:2
        xx(t-1) = floor(NextState/(N/2));
        NextState = Trellis( (NextState+1), t);
    end

    if (ZeroTail)
        xx = xx(1:end-K+1);
    end
    % ************************end of file*************************************
```

```
% ************************beginning of file*****************************
%ReceiverHD.m
function [DataOut, Metric] = ReceiverHD(HDchips, G, Scrambler);
%
% 此函数用于实现基于 Viterbi 译码的硬判决接收机

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% SDchips           硬判决 RAKE 接收机输入符号
% G                 Viterbi 编码生成多项式矩阵
% Scrambler         扰码序列
% DataOut           接收数据（二进制形式）
% Metric            Viterbi 译码最佳度量
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if (nargin == 1)
  G = [1 1 1 1 0 1 0 1 1; 1 0 1 1 1 0 0 0 1];
end

% 速率=19.2 KBps
HDchips = xor(HDchips, Scrambler);

INTERL = reshape(HDchips, 16, 24);
HDchips = reshape(INTERL', length(HDchips), 1);

[DataOut Metric] = VitDec(G, HDchips, 1);
%************************end of file*********************************


% ************************beginning of file*****************************
%VitDec.m
function [xx, BestMetric] = VitDec(G, y, ZeroTail);

%
% 此函数是实现硬判决输入的 Viterbi 译码

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% G             生成多项式的矩阵
% y             输入的待译码序列
% ZeroTail      判断是否包含‘0’尾
% xx            Viterbi 译码输出序列
% BestMetric    最后的最佳度量
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

L = size(G, 1);      % 输出码片数
K= size(G, 2);       % 生成多项式长度
N = 2^(K-1);         % 状态数
T = length(y)/L;     % 最大栅格深度

OutMtrx = zeros(N, 2*L);
```

```
    for s = 1:N
        in0 = ones(L, 1)*[0, (dec2bin((s-1), (K-1))-'0')];
        in1 = ones(L, 1)*[1, (dec2bin((s-1), (K-1))-'0')];

        out0 = mod(sum((G.*in0)'), 2);
        out1 = mod(sum((G.*in1)'), 2);

        OutMtrx(s, :) = [out0, out1];
    end

    PathMet = [0; 100*ones((N-1), 1)];
    PathMetTemp = PathMet(:,1);

    Trellis = zeros(N, T);
    Trellis(:,1) = [0 : (N-1)]';

    y = reshape(y, L, length(y)/L);
    for t = 1:T

        yy = y(:, t)';
        for s = 0:N/2-1
            [B0 ind0] = min(  PathMet(1+[2*s, 2*s+1]) + [sum(abs(OutMtrx(1+2*s,
0+[1:L]) - yy).^2); sum(abs(OutMtrx(1+(2*s+1), 0+[1:L]) - yy).^2)] );
            [B1 ind1] = min(  PathMet(1+[2*s, 2*s+1]) + [sum(abs(OutMtrx(1+2*s,
L+[1:L]) - yy).^2); sum(abs(OutMtrx(1+(2*s+1), L+[1:L]) - yy).^2)] );

            PathMetTemp(1+[s, s+N/2]) =  [B0; B1];
            Trellis(1+[s, s+N/2], t+1) = [2*s+(ind0-1); 2*s + (ind1-1)];
        end
       PathMet = PathMetTemp;

    end

    xx = zeros(T, 1);
    if (ZeroTail)
        BestInd = 1;
    else
        [Mycop, BestInd]  = min(PathMet);
    end

    BestMetric = PathMet(BestInd);
    xx(T) = floor((BestInd-1)/(N/2));

    NextState = Trellis(BestInd, (T+1));
    for t=T:-1:2
        xx(t-1) = floor(NextState/(N/2));
        NextState = Trellis( (NextState+1), t);
    end

    if (ZeroTail)
        xx = xx(1:end-K+1);
```

```
end
% ************************end of file*********************************
```

12.4 本章小结

本章首先介绍了 IS-95 标准，然后分析了 IS-95 前向链路系统各个模块的设计，最后实现了 IS-95 前向链路系统的 MATLAB 程序仿真。读者学习的时候，请注意仿真参数的设置，以保证最终理想的仿真效果。

第 13 章　OFDM 通信系统仿真设计

OFDM 的全称为 Orthogonal Frequency Division Multiplexing，意为正交频分复用。OFDM 的思想可以追溯到 20 世纪 60 年代，当时人们对多载波调制做了许多理论上的工作，论证了在存在符号间干扰的带限信道上采用多载波调制可以优化系统的传输性能；1970 年 1 月，有关 OFDM 的专利被首次公开发表；1971 年，Weinstein 和 Ebert 在 IEEE 杂志上发表了用离散傅里叶变换实现多载波调制的方法；20 世纪 80 年代，人们对多载波调制在高速调制解调器、数字移动通信等领域中的应用进行了较为深入的研究，但是由于当时技术条件的限制，多载波调制没有得到广泛的应用；进入 20 世纪 90 年代，由于数字信号处理技术和大规模集成电路技术的进步，OFDM 技术在高速数据传输领域受到了人们的广泛关注。现在 OFDM 已经在欧洲的数字音视频广播（如 DAB 和 DVB）、欧洲和北美的高速无线局域网系统（如 HIPERLAN2、IEEE 802.11a）、高比特率数字用户线（如 ADSL、VDSL）以及电力线载波通信（PLC）中得到了广泛的应用。

OFDM 通信技术是多载波传输技术的典型代表。多载波传输把数据流分解为若干个独立的子比特流，每个子数据流将具有低得多的比特速率，用这样低比特率形成的低速率多状态符号去调制相应的子载波，就构成了多个低速率符号并行发送的传输系统。OFDM 是多载波传输方案的实现方式之一，利用快速傅里叶逆变换（IFFT，Inverse Fast Fourier Transform）和快速傅里叶变换（FFT，Fast Fourier Transform）来分别实现调制和解调，是实现复杂度最低、应用最广的一种多载波传输方案。

13.1 OFDM 系统的基本原理

13.1.1　正交调制解调

OFDM 是一种多载波调制技术，其原理是用 N 个子载波把整个信道分割成 N 个子信道，即将频率上等间隔的 N 个子载波信号调制并相加后同时发送，实现 N 个子信道并行传输信息。这样每个符号的频谱只占用信道带宽的 $1/N$，且使各子载波在 OFDM 符号周期 T 内保持频谱的正交性。

如图 13-1（a）所示为一个 OFDM 符号内包含 5 个子载波的实例。其中，所有的子载波都具有相同的幅值和相位，但在实际应用中，经过数字基带调制后，每个子载波不可能都有相同的幅值和相位。从图 13-1（a）中可以看出，每个子载波在一个 OFDM 符号周期内都包含整数倍个周期，而且各个相邻的子载波之间相差 1 个周期。这一特性可以用来解释子载波间的正交性，即满足：

$$\frac{1}{T}\int_0^T \mathrm{e}^{\mathrm{j}\omega_n t}\cdot \mathrm{e}^{-\mathrm{j}\omega_m t}\mathrm{d}t=\begin{cases}1, & n=m\\ 0, & n\neq m\end{cases} \qquad (式\ 13\text{-}1)$$

这种正交性还可以从频域角度来解释，图 13-1（b）给出了互相覆盖的各个子信道内经过矩形波成形得到的符号 sinc 函数频谱。每个子载波频率最大值处，所有其他子信道的频谱值恰好为零。因为在对 OFDM 符号进行解调的过程中，需要计算这些点上所对应的每个子载波频率的最大值，所以可以从多个互相重叠的子信道符号中提取每一个子信道符号，而不会受到其他子信道的干扰。从图 13-1（b）中可以看出，OFDM 符号频谱实际上可以满足奈奎斯特准则，即多个子信道频谱之间不存在互相干扰。因此这种一个子信道频谱出现最大值而其他子信道频谱为零的特点可以避免载波间干扰（ICI）的出现。

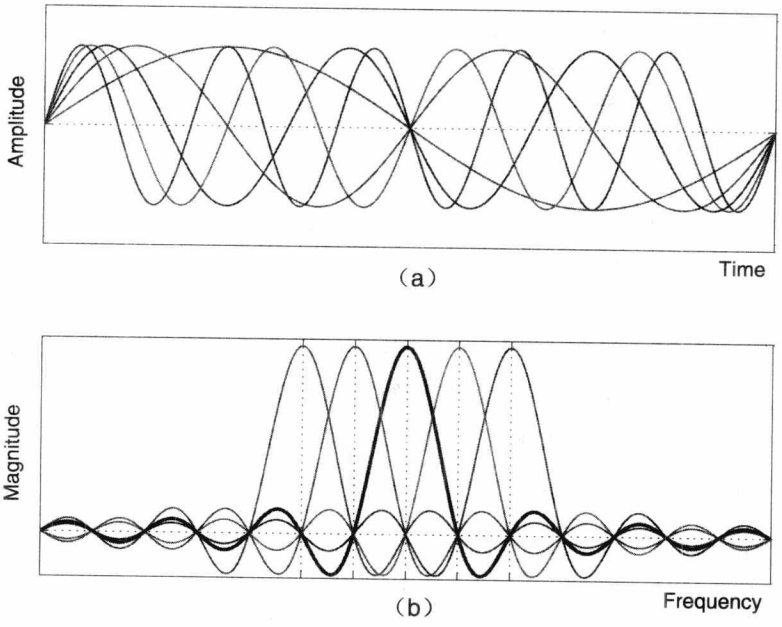


图 13-1　（a）OFDM 子载波时域图（b）OFDM 子载波频域图

在发送端，串行码元序列经过数字基带调制、串并转换，将整个信道分成 N 个子信道。N 个子信道码元分别调制在 N 个子载波频率 $f_0,f_1,\cdots,f_n,\cdots,f_{N-1}$ 上，设 f_c 为最低频率，相邻频率相差 $1/N$，则 $f_n=f_c+n/T$，$n=0,1,2,\cdots,N-1$，角频率为 $\omega_n=2\pi f_n$，$n=0,1,2,\cdots,N-1$。

待发送的 OFDM 信号 $D(t)$ 为：

$$D(t)=\mathrm{Re}\left(\sum_{n=0}^{N-1}X(n)\cdot \mathrm{e}^{\mathrm{j}\omega_n t}\right)=\cos 2\pi f_c t\cdot \mathrm{Re}\left(\sum_{n=0}^{N-1}X(n)\cdot \mathrm{e}^{\mathrm{j}2\pi nt/T}\right)$$
$$-\sin 2\pi f_c t\cdot \mathrm{Im}\left(\sum_{n=0}^{N-1}X(n)\cdot \mathrm{e}^{\mathrm{j}2\pi nt/T}\right),\quad t\in[0,T] \qquad (式\ 13\text{-}2)$$

接收端对接收到的信号进行如下解调：

$$X'(m)=\frac{1}{T}\int_0^T D(t)\cdot e^{-j2\pi f_m t}dt=\frac{1}{T}\int_0^T\sum_{n=0}^{N-1}X(n)\cdot e^{j\omega_n t}\cdot e^{-j\omega_m t}dt$$
$$=\sum_{n=0}^{N-1}X(n)\cdot\frac{1}{T}\int_0^T e^{j\omega_n t}\cdot e^{-j\omega_m t}dt,\quad t\in[0,T] \qquad (式\ 13\text{-}3)$$

由于 OFDM 符号周期T内各子载波是正交的，正交关系如式 13-1 所示。所以，当$n=m$时，调制载波ω_n与解调载波ω_m为同频载波，满足相干解调的条件，$X'(m)=X(m)$，$m=0,1,2,\cdots,N-1$，恢复了原始信号；当$n\neq m$时，接收到的不同载波之间互不干扰，无法解调出信号。这样就在接收端完成了信号的提取，实现了信号的传输。

在式 13-2 中，设

$$y(t)=\sum_{n=0}^{N-1}X(n)\cdot e^{j2\pi nt/T},\quad t\in[0,T] \qquad (式\ 13\text{-}4)$$

若 1 个T内$y(t)$以采样频率$f_s=1/\Delta t$（其中$1/\Delta t=T/N$）被采样，则可得N个采样点。设$t=k\Delta t$，$nt/T=nk/N$，则

$$y(k)=\sum_{n=0}^{N-1}X(n)\cdot e^{j2\pi nk/N},\quad k=0,1,2,\cdots,N-1 \qquad (式\ 13\text{-}5)$$

式 13-5 正是序列$\{X(n),\ n=0,1,2,\cdots,N-1\}$的$N$点离散傅里叶反变换（IDFT）的结果，这表明 IDFT 运算可完成 OFDM 基带调制过程。而其解调过程可通过离散傅里叶变换（DFT）实现。因此，OFDM 系统的调制和解调过程等效于 IDFT 和 DFT。在实际应用中，一般用 IFFT/FFT 来代替 IDFT/DFT，这是因为 IFFT/FFT 变换与 IDFT/DFT 变换的作用相同，并且有更高的计算效率，适用于所有的应用系统。

13.1.2 系统组成

OFDM 系统组成框图如图 13-2 所示。其中，上半部分对应于发射机链路，下半部分对应于接收机链路，整个系统包含信道编/解码、数字调制/解调、IFFT/FFT、加/去保护间隔和数字上/下变频。

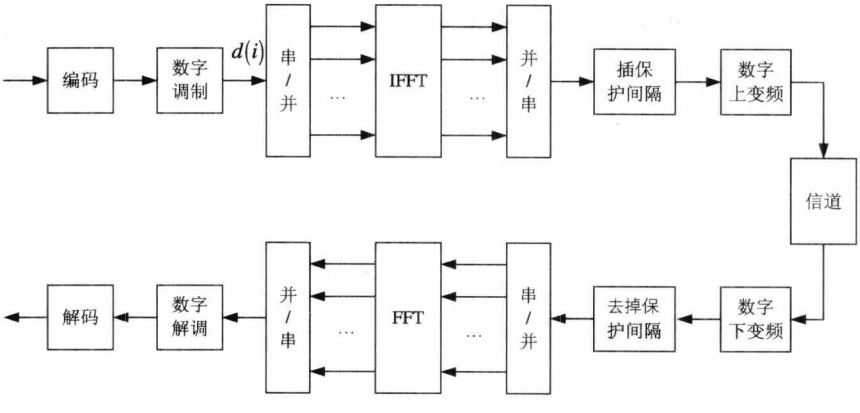


图 13-2 OFDM 系统组成框图

输入比特序列完成信道编码后，根据采用的调制方式，完成相应的调制映射，形成调制信息序列$\{X(n)\}$，对$\{X(n)\}$进行 IFFT，将数据的频谱表达式变换到时域上，得到 OFDM 已调信号的时域抽样序列，加上保护间隔（通常采用添加循环前缀的方式），再进行数字变频，得到 OFDM 已调信号的频带时域波形。接收端先对接收信号进行数字下变频，去掉保护间隔，得到 OFDM 已调信号的抽样序列，对该抽样序列做 FFT 即得到原调制信息序列$\{X(n)\}$。

1．信道编码

为了提高数字通信系统的性能，信道编码（通常还伴有交织）是普遍采用的方法。在 OFDM 系统中，如果信道衰落不是太严重，均衡是无法再利用信道的分集特性来改善系统性能的，因为 OFDM 系统自身具有利用信道分集特性的能力，一般的信道特性信息已经被 OFDM 这种调制方式本身所利用了。但是，OFDM 系统的结构却为在子载波间进行编码提供了机会，形成 COFDM（前置编码 OFDM）方式。编码可以采用各种码，如分组码、卷积码等，其中卷积码的效果要比分组码好，但分组码的编解码实现更为简单。

2．子载波调制

传输信号进行信道编码后，要进行子载波的数字调制将其转换成载波幅度和相位的映射，一般采用 QAM 或 MPSK 方式。各子载波不必要采用相同的状态数（进制数），甚至不必要采用相同的调制方式。这使得 OFDM 支持的传输速率可以在一个较大的范围内变化，并可以根据子信道的干扰情况，在不同的子信道上采用不同状态数的调制，甚至采用不同的调制方式。调制信号星座在 IFFT 之前根据调制模式形成。

3．保护间隔

应用 OFDM 的一个重要原因在于它可以有效地对抗多径时延扩展。把输入数据流串并变换到 N 个并行的子信道中，使得每一个调制子载波的数据周期可以扩大为原始数据符号周期的 N 倍，因此时延扩展与符号周期的数值比也同样降低 N 倍。另外，通过在每个 OFDM 符号间插入保护间隔（GI，Guard Interval）可以进一步抵制符号间干扰（ISI），还可以减少在接收端的定时偏移错误。这种保护间隔是一种循环复制，增加了符号的波形长度，在符号的数据部分，每一个子载波内有一个整数倍的循环，此种符号的复制产生了一个循环的信号，即将每个 OFDM 符号的后 T_g 时间中的样点复制到 OFDM 符号的前面，形成循环前缀（CP，Cyclic Prefix），在交接点没有任何的间断。因此将一个符号的尾端复制并补充到起始点增加了符号时间的长度。

图 13-3 为循环前缀示意图，并进一步说明了多径传播对 OFDM 符号所造成的影响，图中主径表示第一条路径到达的信号，多径干扰信号表示其他路径到达的实线信号的时延信号。实际上，OFDM 接收机所能看到的只是所有这些信号之和，但是为了更加清楚地说明多径的影响，还是分别给出了每个子载波信号。

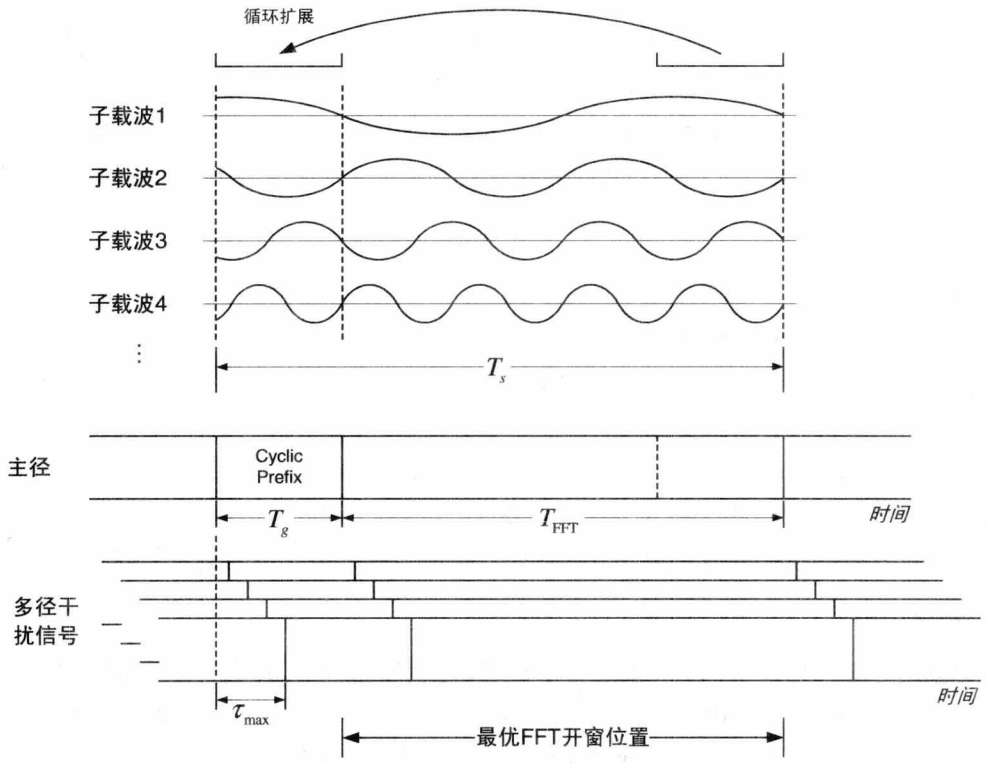


图 13-3　OFDM 系统中保护间隔的添加

符号的总长度为 $T_s = T_g + T_{FFT}$，其中 T_s 为 OFDM 符号的总长度，T_g 为抽样的保护间隔长度，T_{FFT} 为 FFT 变化产生的无保护间隔的 OFDM 符号长度，则在接收端抽样开始的时刻 T_x 应该满足下式：

$$\tau_{\max} < T_x < T_g \qquad (式\ 13\text{-}6)$$

其中 $\tau_{\max}$ 是信道的最大多径时延扩展，当抽样满足式 13-6 时，由于前一个符号的干扰只会存在于 $[0, \tau_{\max}]$，所以当子载波个数比较大时，OFDM 的符号周期 T_s 相对于信道的脉冲响应长度 $\tau_{\max}$ 很大，则 ISI 的影响很小，甚至会没有 ISI；而如果相邻 OFDM 符号之间的保护间隔 T_g 满足 $T_g > \tau_{\max}$ 的要求，则可以完全克服 ISI 的影响。同时，由于 OFDM 延时副本内所包含的子载波的周期个数也为整数，时延信号就不会破坏子载波间的正交性，在 FFT 解调过程中就不会产生载波间干扰（ICI）。

4．数字上下变频

OFDM 调制器的输出产生了一个基带信号，发射机将此基带信号与所需传输的频率进行上变频操作，接收机需要对中频进行接收，之后进行 OFDM 基带解调。上下变频部分可由模拟技术或数字技术完成，两种技术虽然完成同样的操作，但是由于数字调制技术提高了 I、Q 信道间的匹配性和数字 I、Q 调制器相位准确性，将会使混频结果更精确。另外，上下变频中通常伴有基带成形滤波器和采样率转换器等，采用数字技术更利于实现。

13.1.3　OFDM 的优点

1．频谱利用率较高

OFDM 技术可以被看作是一种调制技术，也可以被当作一种复用技术。传统的频分复用（FDM）多载波调制技术（如图 13-4（a）所示）中各个子载波的频谱是互不重叠的，同时，为了减少各子载波之间的相互干扰，子载波之间需要保留足够的频率间隔，频谱利用率较低；而 OFDM 多载波调制技术（如图 13-4（b）所示）中各子载波的频谱是互相重叠的，并且在整个符号周期内满足正交性，不但减小了子载波间的相互干扰，还大大减少了保护带宽，提高了频谱利用率。

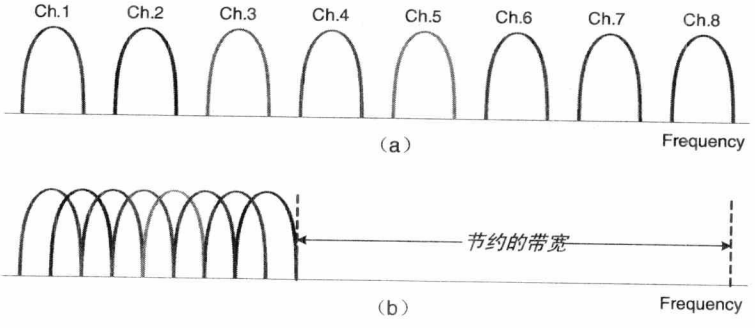


图 13-4　（a）FDM 调制技术（b）OFDM 调制技术

2．抗码间干扰（ISI，Inter-Symbol Interference）能力强

码间干扰是数字通信系统中除噪声干扰之外最主要的干扰，它与加性的噪声干扰不同，是一种乘性的干扰。造成码间干扰的原因有很多，实际上，只要传输信道的频带是有限的，就会造成一定的码间干扰。OFDM 通过在传输的数据块之间插入一个大于信道脉冲响应时间的保护间隔，消除了由于多径时延扩展引起的符号间干扰。

3．抗频率选择性衰落和窄带干扰能力强

在单载波系统中，一次衰落或者干扰会导致整个链路失效，但是在多载波系统中，某一时刻只会有少部分的子信道受到深衰落的影响。OFDM 把信息通过多个子载波传输，在每个子载波上的信号时间就相应地比同速率的单载波系统上的信号时间长很多倍，使 OFDM 对脉冲噪声和信道快速衰落的抵抗力更强。同时，通过子载波的联合编码，达到了子信道间的频率分集的作用，也增强了对脉冲噪声和信道快速衰落的抵抗力。OFDM 还可以根据每个子载波的信噪比来优化分配每个子载波上传送的信息比特，自动控制各个子载波的使用，有效避开噪声干扰以及频率选择性对数据传输可靠性的影响，实现对信道的自适应性。通过软件编程，OFDM 可以有效地屏蔽某些子载波，实现对民用或军用重要频点的保护。在电力线通信中，OFDM 通过把电力线分为许多窄带子信道，使得各个子信道呈现相对性和平坦特性，不仅消除了由于电力线的低通效应和传递函数的剧烈波动而引起的失真，而且无须复杂的信道均衡系统，实现比较简单，成本比较低廉。

13.1.4 OFDM 的缺点

由于 OFDM 系统存在多个正交的子载波，而且其输出信号是多个子信道的叠加，因此与单载波系统相比，存在如下缺点：

（1）易受频率偏差的影响。

由于子信道的频谱相互覆盖，这就对它们之间的正交性提出了严格的要求。在传输过程中出现的信号频谱偏移或发射机与接收机本地振荡器之间存在频率偏差，都会使 OFDM 系统子载波之间的正交性遭到破坏，导致子信道间干扰（ICI，Inter-Channel Interference），这种对频率偏差的敏感性是 OFDM 系统的主要缺点之一。

（2）存在较高的峰值平均功率比。

多载波系统的输出式多个子信道信号的叠加，因此如果多个信号的相位一致时，所得到的叠加信号的瞬时功率就会远远高于信号的平均功率，导致较大的峰值平均功率比（PAPR，Peak-to-Average Power Ratio）。这就对发射机内放大器的线性度提出了很高的要求，因此可能带来信号畸变，使信号的频谱发生变化，从而导致各个子信道间的正交性遭到破坏，产生干扰，使系统的性能恶化。

13.1.5 OFDM 的关键技术

1．时域和频域同步

OFDM 块是由保护间隔和有用数据信息组成，因此 OFDM 中的定时同步就是要确定 OFDM 块有用数据信息的开始时刻，也可以叫做确定 FFT 窗的开始时刻。定时的偏移会引起子载波相位的旋转，而且相位旋转角度与子载波的频率有关，频率越高，旋转角度越大。如果定时的偏移量与最大时延扩展的长度之和大于循环前缀的长度，这时一部分数据信息丢失了，而且最为严重的是子载波之间的正交性被破坏了，由此带来了 ISI 和 ICI，这是影响系统性能的关键问题之一。

频率偏移是由收发设备的本地载频之间的偏差、信道的多普勒频移等引起的，由子载波间隔的整数倍偏移和子载波间隔的小数倍偏移构成。频率偏移破坏了子载波间的正交性，导致子载波之间产生干扰。

OFDM 中的同步算法有很多种，目前，OFDM 系统中的定时同步主要解决方法有循环前缀法、PN 前缀法和特殊训练符号法等，频偏估计的方法有最大似然估计法等。

2．降低峰值平均功率比

由于 OFDM 信号时域上表现为 N 个正交子载波信号的叠加，当这 N 个信号恰好均以峰值相加时，OFDM 信号也将产生最大峰值（如图 13-5 所示），该峰值功率是平均功率的 N 倍。尽管峰值功率出现的概率较低，但为了不失真地传输这些高峰值平均功率比（PAPR）的 OFDM 信号，发送端对高功率放大器（HPA）的线性度要求很高，从而导致发送效率极低，接收端对前端放大器以及 A/D 转换器的线性度要求也很高。因此，高的 PAPR 使得 OFDM 系统的性能大大下降甚至直接影响实际应用。目前，已有很多文献讨论了 OFDM

的降低 PAPR 的算法，这些方法主要有 3 类：信号畸变技术、编码方法（包括分组码、格雷互补码和多相互补序列等）和基于信号空间扩展的方法。

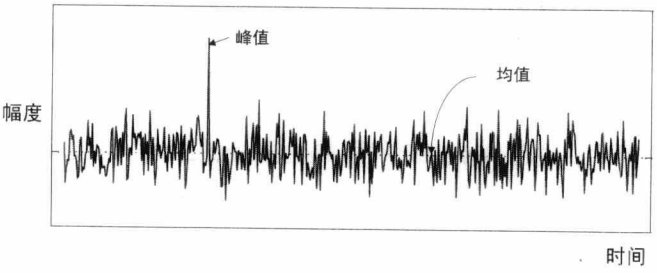


图 13-5　存在 PAPR 问题的 OFDM 信号，$N=512$

3. 信道编码

在无线衰落环境下，如果不采用适当的前向纠错编码技术，要想得到满意的差错性能几乎是不可能的。在实际信道上传输数字信号时，为了克服信道特性不理想及加性噪声的影响，首先应该考虑合理设计基带信号、选择调制解调方式、采用时域频域均衡等技术使误比特率降低。进一步应该采用差错控制编码技术来降低误比特率以满足系统指标要求。

例如脉冲噪声的存在产生的错误往往是突发错误或突发错误与随机错误并存。为了纠正比较长的突发错误，或者利用码的纠随机错误能力来纠正突发错误，常常使用交织技术。采用交织方法构造出来的码称为交织码。交织的作用是减小信道中错误的相关性，把长的突发错误离散成短的突发错误或随机错误。交织深度越大，则离散程度越高。在系统中，可以从时域和频域两个角度使用来对抗频率选择性衰落和时间选择性衰落。为了达到这个目的，通常使用的一种技术是交织编码技术。近几年兴起了若干新型编码技术，比如 Turbo 码、网格编码技术、空时编码技术等，也都在系统中得到应用。

13.2 OFDM 系统的 PAPR 抑制算法设计

13.2.1　OFDM 信号的 PAPR 及其分布

与任何多载波调制系统一样，OFDM 也面临着峰均功率比过大的问题。对于一个 OFDM 系统而言，由于复合包络是多个子载波信号的叠加，所以它将会有大的包络变化范围，因此会产生很大的 PAPR（相对于单载波系统而言）。通常，PAPR 与子载波数 N 之间呈现正比的关系。因此，在 OFDM 技术日益得到广泛应用的今天，很多学者正在致力于研究如何找出一套合理的理论和方法，来降低 OFDM 系统中所存在的高峰均比问题。中心极限理论阐述了独立同分布的、均值为零的随机变量，在变量数据量趋向于无穷时，其线性组合可以近似看作是一种均值为零的高斯分布。对于 OFDM 信号而言，一般当子载波数 $N \geqslant 64$ 时就认为符合上述规律。在 OFDM 中，实际发射的信号是多个子载波信号的叠加，这将不可避免地导致信号的包络变化非常剧烈，如果 N 个子载波的信号均以相同的相位相加时，就会产生一个 OFDM 信号的峰值功率，这个峰值功率是平均功率的 N 倍，

也就是说，最大峰值功率与平均功率的比值为 N。通常，我们将在一段时间内最大峰值功率与平均功率的比值称为峰值平均功率比（Peak-to-Average Power Ratio，PAPR）。当子载波数很大时，这种剧烈的发射功率变化对射频放大器的设计提出了很高的要求，阻碍 OFDM 技术的实际应用。因此在 OFDM 系统中，PAPR 的分析和降低就变得尤为重要。

1. PAPR 的定义

与单载波系统相比，由于 OFDM 符号是由多个独立的、经过调制的子载波信号相加而成的，这样的合成信号就有可能产生比较大的峰值功率，由此会带来较大的峰值平均功率比，简称峰均比。OFDM 系统中峰均比的定义为：

$$\mathrm{PAPR(dB)}=10\lg\frac{\max\left\{|x_n|^2\right\}}{E\left\{|x_n|^2\right\}} \qquad (式\ 13\text{-}7)$$

其中，x_n 表示经过 IFFT 运算之后所得到的输出信号，即

$$x_n=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X_kW_N^{nk} \qquad (式\ 13\text{-}8)$$

对于包含 N 个子信道的 OFDM 系统来说，当 N 个子信号都以相同的相位求和时，所得到信号的峰值功率就会是平均功率的 N 倍，因而基带信号的峰均比可以为：PAPR = 10log10N，例如 $N=256$ 的情况中，OFDM 系统的 PAPR = 24dB，当然这是一种非常极端的情况，OFDM 系统内的峰均比通常不会达到这一数值。图 13-6 以 $N=16$ 为实例，说明了 OFDM 系统中存在较大 PAPR 的这种现象。在这个实例中，所有子载波都受到相同初始相位的符号的调制。通过该实例可以看到：峰值功率是平均功率的 16 倍。对于未经过调制的载波来说，其 PAPR = 0dB。

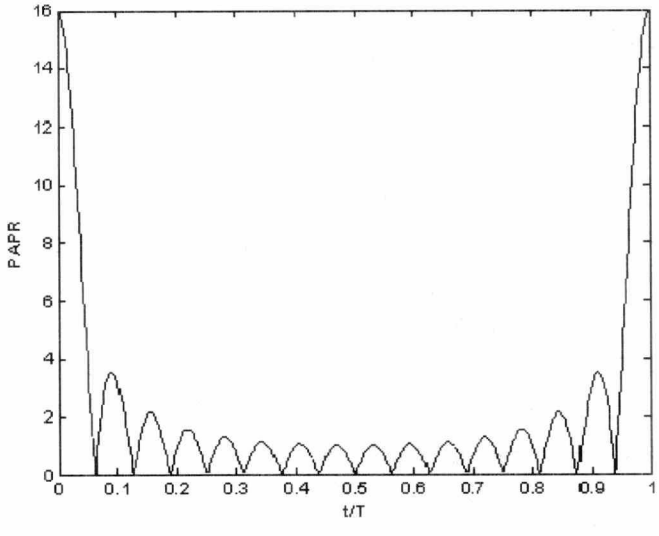


图 13-6　OFDM 信号存在 PAPR=16 的情况

2. PAPR 的统计特性

对于包含 N 个子载波的 OFDM 系统来说，经过 IFFT 计算得到的功率归一化的复基带符号是：

$$x(t)=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X_k\,\mathrm{e}^{\mathrm{j}k\Delta ft} \qquad (式\ 13\text{-}9)$$

其中，X_k 表示第 k 个子载波上的调制符号。例如，对于 QPSK 调制来说，$X_k\in\{1,-1,j,-j\}$。根据中心极限定理，对于较大子载波数 N，信号 $x(t)$ 的实部和虚部的样点都服从均值为 0、方差为 0.5 的高斯分布，因此，OFDM 符号的幅度服从瑞利分布，功率服从有中心的、两个自由度的 χ^2 分布（均值为 0，方差为 1），其累积分布函数为 $p_{power}(y)=e^{-y}$，所以，可以得到其累积分布函数（CDF）为：

$$P(\mathrm{power},z)=\int_0^z e^{-y}\mathrm{d}y=1-\exp(-z) \qquad (式\ 13\text{-}10)$$

假设 OFDM 符号周期内每个采样值之间是不相关的，则在 OFDM 符号周期内的 N 个采样值当中，每个样值的 PAPR 都小于门限值 z 的概率分布为：

$$P\{\mathrm{PAPR},z\}=(1-e^{-z})^N \qquad (式\ 13\text{-}11)$$

对 OFDM 符号周期内进行过采样有助于更加准确地反映符号的变化情况，特别是针对 PAPR 而言，由于最后送到放大器中的应该是经过 D/A 变换的连续信号，因此过采样更加有助于收集到较大的峰值功率，从而可以更加准确地衡量 OFDM 系统内的 PAPR 特性。所以，对 OFDM 符号实施过采样是非常必要的，但是这样做会使采样符号之间的非相关性遭到破坏，也就是说，使采样符号之间存在一定的相关性。但是如果基于符号之间的相关性来考虑 PAPR 的准确表达式比较困难，就可以假设利用对 αN 个子载波进行非过采样来近似描述对 N 个子载波的过采样，其中 $\alpha>1$。因此，对 OFDM 符号实施过采样，就可以看作添加一定数量相互独立的样本值。PAPR 的概率分布可以表示为

$$P\{\mathrm{PAPR}\leqslant z\}=(1-e^{-z})^{\alpha N} \qquad (式\ 13\text{-}12)$$

实施过采样可以更加准确地反映 OFDM 系统内 PAPR 的分布情况，而且当 $N\geqslant 64$ 时，上式比较能够反映真实的状况。或者，可以从另一个角度来衡量 OFDM 系统的 PAPR 分布，即计算峰均比超过某一门限值 z 的概率，得到互补积累分布函数 CCDF：

$$P\{\mathrm{PAPR}>z\}=1-P\{\mathrm{PAPR}\leqslant z\}=1-(1-e^{-z})^N \qquad (式\ 13\text{-}13)$$

CCDF 曲线是 x 的平滑非递增函数，体现了信号功率高于给定功率电平的统计情况。它的 X 坐标表示信号峰值功率高出平均功率的 dB 电平值，Y 坐标表示当信号峰值功率大于或等于 X 坐标所指定的某一功率电平时所占用的时间比率。在随后的讨论中，我们采用互补累积分布函数（CCDF）来衡量 OFDM 系统中的 PAPR 分布。

3. 高 PAPR 产生的原因及问题

OFDM 系统中产生高 PAPR 的主要原因是 OFDM 信号在时域上表现为 N 个正交子载波的叠加，当子载波个数达到一定程度后，根据中心极限定理，OFDM 符号的波形将是一个高斯随机过程，其包络具有不稳定性，当这 N 个子载波恰好均以峰值点相加时将产生最大的峰值，从而形成高的 PAPR。这种现象将导致 OFDM 信号通过放大器时容易受到非线性失真，破坏子载波之间的正交性，从而恶化传输性能。对多载波系统而言，峰均比主要取决于子载波的个数，随着子载波个数的增加而增加。高 PAPR 带来最严重的影响是在发射端和接收端的功率放大器上。由于一般的功率放大器都不是线性的，而且其动态范围也是有限的，所以当 OFDM 系统内这种变化范围较大的信号通过非线性部件（例如进入放大器的非线性区域）时，信号会产生非线性失真，产生谐波，造成较明显的频谱扩展干扰以及带内信号畸变，导致整个系统性能下降，而且同时还会增加 A/D 和 D/A 转换器的复杂度并且降低它们的准确性。AM/AM 放大器的一般模型表示为：

$$O(x)=\frac{x}{(1+x^{2p})^{1/2p}} \qquad \text{（式 13-14）}$$

在现有的实用放大器中，p 的取值范围一般介于 2 到 3 之间。对于较大的 p 值来说，可以近似地被看作限幅器，即只要小于最大输出值，该放大器就是线性的，一旦超过了最大输出门限值，则对该峰值信号进行限幅。因此 PAPR 较大是 OFDM 系统所面临的一个问题，所以必须要考虑如何减少大峰值功率信号的出现概率，从而避免非线性失真的出现。

13.2.2 降低 PAPR 的常用方法

目前，降低 OFDM 信号 PAPR 的方法很多，大体可以分成三大类：信号预畸变技术、编码类技术和概率类技术。这三种方法各有特色和着眼点，但每类方法都存在着缺陷。信号预畸变技术直接对信号的峰值进行非线性操作，它最直接，最简单，但会带来带内噪声和带外干扰，从而降低系统的误比特率性能和频谱效率。编码类技术利用编码将原来的信息码字映射到一个具有较好 PAPR 特性的传输码集上，从而避开了那些会出现信号峰值的码字。该类技术为线性过程，它不会使信号产生畸变。但是，编码类技术的技术复杂度非常高，编解码都比较麻烦。更重要的是，这类技术的信息速率降低得很快，因此只适用于子载波数比较少的情况。概率类技术不像编码类技术那样完全避开信号的峰值，而是着眼于努力降低信号峰值出现的概率。该类技术采用的方法也为线性过程，因此，它不会对信号产生畸变。这类技术能够很有效地降低信号的 PAPR 值，它的缺点在于计算复杂度太大。

下面就常见的几种算法做简要介绍。

1. 信号预畸变

信号预畸变技术包括限幅类技术和压缩扩张变换。

（1）限幅

限幅是最简单的方法，它采用非线性过程，直接在 OFDM 信号幅度峰值或附近采用

非线性操作来降低信号的 PAPR 值，能适用于任何数目子载波构成的系统。限幅相当于对原始信号加一矩形窗，如果 OFDM 信号的幅值小于预先给定的门限值时，该矩形窗函数的幅值就为 1，否则，矩形窗函数的幅值就小于 1。可见，限幅会不可避免地产生信号畸变。由于存在信号的失真（信号有所畸变），因而限幅法不可避免地产生一种自干扰，从而必然造成系统 BER 性能的下降。其次，限幅还会因为信号的非线性畸变导致带外频谱的辐射或称为频谱泄露（带外辐射功率的增大），虽然带外频谱的辐射可以通过应用非矩形的窗函数来解决（如 Gaussian、Kaiser 和 Hamming 窗等），但效果都不是很明显。

（2）压缩扩张变换

它是借用语音处理中基于 μ 律非均匀量化的一种非线性变换函数，实现起来非常简单，计算复杂度也不会随着子载波数的增加而增加。压缩扩张变换主要是对较小幅值信号的功率进行放大，而保持较大幅值信号的功率不变，以增大整个系统的平均功率为代价来达到降低 PAPR 的目的，因而其弊端在于：一方面系统的平均发射功率要增大；另一方面使得符号的功率值更加接近高功率放大器的非线性变化区域，造成了信号的失真。

2. 编码类技术

编码类技术主要是利用不同编码所产生不同的码组而选择 PAPR 较小的码组作为 OFDM 符号进行数据信息的传输，从而避免了信号峰值，此类技术为线性过程，不会使信号产生畸变，但其计算复杂度非常高，编解码都比较复杂，而且信息速率降低很快，因此，只适用于子载波数比较少的情况。其主要方法有：分组编码法（Block Coding）、格雷补码序列（Golay Complementary Sequences，GCS）和雷德密勒（Reed-Muller）码等。

基于分组编码降低 OFDM 系统 PAPR 方法的基本思想是：在对比特流进行 IFFT 运算之前，先进行特殊的编码处理（如应用奇偶校验位），使得输出的比特流经过 OFDM 调制后具有较低的 PAPR。精心设计的分组编码方法不仅可以有效地降低 PAPR，同时还可以起到类似于信道编码的作用，使系统具有前向检错和纠错的能力。

应用格雷互补序列的方法就是把 GCS 作为 IFFT 的输入，那么其输出信号就会有比较低的 PAPR 值，并且在时/频域中具有较好的信道估计和纠错能力。应用 GCS 序列对，其最大的优点就是不论子载波数多少，其 PAPR 可以降到 3dB 以内。但是，由于子载波数目的逐渐增多，寻找最佳生成矩阵和相位旋转向量的难度显著上升，因而目前的 GCS 法并不适用于子载波数很多的 OFDM 系统。

应用编码方法降低 PAPR 的优点是系统相对简单、稳定，降低 PAPR 的效果好。但是，它的缺点也非常明显，一是受编码调制方式的限制，比如分组编码只适用于 PSK 的调制方式，而不适用于基于 QAM 调制方式的 OFDM 系统；二是受限于子载波个数，随着子载波数的增加，计算复杂度增大，系统的吞吐量严重下降，带宽的利用率显著降低；三是数据的编码速率有所减小，这是因为大部分的编码方法都要引入一定的冗余信息。

3. 概率类技术

概率类技术并不着眼于降低信号幅度的最大值，而是降低峰值出现的概率。一般的概率类技术都将带来一定的信息冗余。这类技术主要包括选择映射方法（SLM）及部分序列

传输方法（PTS）。

（1）选择映射（SLM）

SLM 方法的基本思想是用 D 个统计独立的向量 Y_d 表示相同的信息，选择其时域符号 y_d 具有最小 PAPR 值的一路用于传输，SLM 原理如图 13-7 所示。

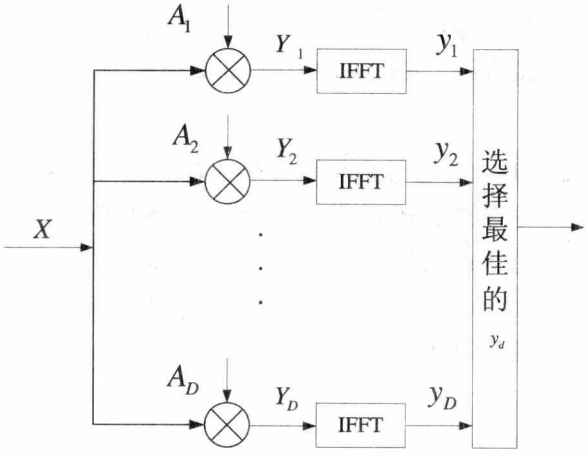


图 13-7　SLM 原理图

其中，D 路相互独立的向量 Y_d 是由 D 个固定的但完全不同的旋转向量 A_d（$1 \leqslant d \leqslant D$）产生的，可以设定第一路信号 Y_1 为原始信号 X，也就是说设定 A_1 为单位向量，这并不会带来任何的性能损失。具体操作过程是，当原始数据向量发送后，所有 D 路并行计算其对应的时域信号 y_d，并选择具有最小 PAPR 值的一路进行传送。由于其需要 D 个并行的 IFFT 操作，因此，采用该种方法的系统成本比较大。

对 SLM 方法，在接收端必须进行与发送端相反的操作以恢复出传输的原始信息，因此，接收端必须知道发送端选择的是哪一路信号进行传送的。最简单的解决方法是将选择的支路序号 d 作为边带信息一起传送到接收端。由于这种边带信息对接收端正确恢复传送的原始信息至关重要，因此一般采用信道编码以保证其可靠传送。通常对 D 路 SLM 发送机需要传送 $\log_2(D-1)$ 比特的边带信息。

（2）部分传输序列（PTS）

PTS 也是基于 SLM 相同的原理，但其转换向量具有不同的结构。PTS 方法首先将进来的数据向量划分为 V 个互不重叠的子向量 X_v，则每个子向量的长度变为 N/V。由于它们互不重叠，因此有：

$$X = \sum_{v=1}^{V} X_v \qquad \text{（式 13-15）}$$

子向量 X_v 中的每个子载波都乘以相同的旋转因子 $R_d^{(v)}$，不同子向量的旋转因子是统计独立的。这就意味着旋转向量 A_d 只包含 V 个独立的元素。由此有：

$$y_d = IFFT\left(\sum_{v=1}^{V} Y_d^{(v)}\right)$$

$$= \left(\sum_{v=1}^{V} IFFT(Y_d^{(v)})\right) \quad , \quad 1 \leqslant d \leqslant D \qquad (式\ 13\text{-}16)$$

$$= \sum_{v=1}^{V} R_d^{(v)} \bullet IFFT(X_d^{(v)})$$

上式推导利用了 IFFT 的线性性质，这也显示了这种方法的优越性：d 个时域向量 y_d 可以在 IFFT 操作后进行构造，从而每次迭代就不需要再进行 IFFT 操作。

在发送端，具有最小 PAPR 值的信号 y_d 被传送，接收端为了恢复发送端发送的信号，必须知道其传送的信号采用了哪个旋转向量。因此需要额外传送 $(V-1)\log_2 W$ 比特的边带信息。

13.2.3　基于改进脉冲成形技术的 PAPR 抑制方法

脉冲成形技术（PS）的思想是将原始数据序列和成形脉冲矩阵相乘产生新序列，使多载波的各子载波符号间具有一定的相关性，从而改善信号的 PAPR 特性。它只需恰当选择各子载波的时域波形从而避开额外的 IFFT 过程，在有效保持系统带宽效率的情况下，为信道编码留下余地。因此，PS 是一种非常有效的 PAPR 抑制方法。

本节先重点讲述了 PS 技术抑制 OFDM 信号 PAPR 的理论证明，采用了 Nyquist 脉冲成形技术，并仿真验证了该技术的 PAPR 抑制性能和该技术对 OFDM 信号的影响。

1．系统模型

基于 PS 技术的 OFDM 系统发射机原理框图如图 13-8 所示。MPSK 或 MQAM 基带数据序列通过串/并变换后，先分别乘上 N 个成形脉冲，再调制 N 个正交子载波。以 T 表示 OFDM 符号周期，a_n（$n=0,1,\cdots,N-1$）表示每个子载波的调制数据，f_n 表示第 n 个子载波频率，$p_n(t)$ 表示周期为 T，作用于子载波 f_n 的成形脉冲。$0 \leqslant t \leqslant T$ 内 OFDM 复信号表示为：

$$s(t) = \sum_{n=0}^{N-1} a_n p_n(t) \exp(\mathrm{j}2\pi f_n t) \qquad 0 \leqslant t \leqslant T \qquad (式\ 13\text{-}17)$$

其中子载波 $f_n = n/T$。$s(t)$的实部和虚部分别对应于 OFDM 信号的同相和正交分量，在实际系统中可以分别与相应子载波的同相分量和正交分量相乘，合成最终的 OFDM 信号。

PS 中周期为 T 的成形脉冲 $p_n(t)$（$n=0,1,\cdots,N-1$）必须满足下列四个条件：

- 等能量：$\int_0^T |p_n(t)|^2 dt = T$；
- 时限：$p_n(t)=0$，$|t-T/2| > T/2$；

- 带限：$P_n(f-n/T)\approx 0$，$|f-B|>B(1+\beta)$，其中 $P_n(f)$ 为 $p_n(t)$ 的频率响应，$B=1/2T_s$，$T_s=T/N$ 为 Nyquist 采样频率，$0<\beta<1$为与子载波数和发送滤波器相关的系数；
- 正交：$\int_0^T p_m(t)p_n^*(t)\exp[\mathrm{j}2\pi(f_m-f_n)t]\mathrm{d}t=\begin{cases}T, & m=n\\ 0, & m\neq n\end{cases}$。

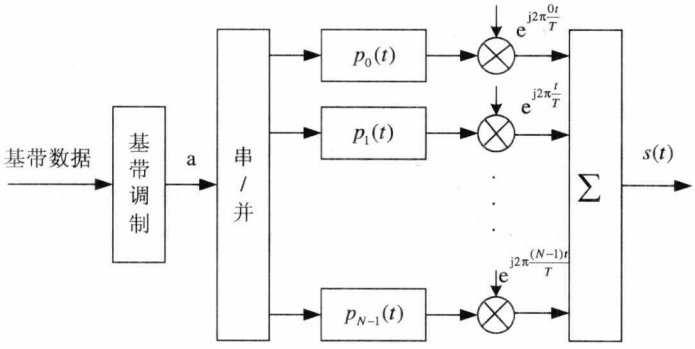


图 13-8　基于脉冲成形技术的 OFDM 发射机原理框图

2. 基于 PS 技术的 PAPR 抑制原理

OFDM 信号的 PAPR 为：

$$\mathrm{PAPR}=\max_{0\leqslant t\leqslant T}|s(t)|^2 / \underset{0\leqslant t\leqslant T}{E}[|s(t)|^2] \qquad (式\ 13\text{-}18)$$

当子载波调制相位一致时，OFDM 信号的峰值将叠加产生很大的峰值功率，导致高 PAPR。如果能够使子载波符号间具有一定的相关性，那么将降低相位一致情况发生的概率，结果是 PAPR 得到抑制。

从 OFDM 符号各采样值的角度出发，考查互相关函数：

$$R_s(t_1,t_2)=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}E[a_n a_m^*]p_n(t_1)p_m{}^*(t_2)\exp[\mathrm{j}2\pi(nt_1-mt_2)/\mathrm{T}] \qquad (式\ 13\text{-}19)$$

由式 13-19 可以看出 OFDM 符号各采样值之间的互相关函数是基带数据和成形脉冲波形的函数。因此，引入采样值间的相关性有两条途径：

（1）引入基带数据间的相关性，也就是通过对输入信息编码来实现。编码方法会不可避免地引入冗余信息，使系统带宽效率降低。

（2）引入子载波波形间的相关性，也就是采用成形脉冲对各子载波进行脉冲成形，它在保持子载波间正交性的同时，不影响系统带宽效率，不需要额外的带外信息。

- 相同成形脉冲

若每个子载波采用相同的成形脉冲波形，即 $p_n(t)=p(t)\ (n=0,1,\cdots,N-1)$，那么式 13-19 可写为：

$$R_s(t_1,t_2)=\sigma^2\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}p_n(t_1)p_m{}^*(t_2)\exp[\mathrm{j}2\pi(nt_1-mt_2)/\mathrm{T}]$$

$$=\begin{cases}\sigma^2N^2p(t_1)p^*(t_2), & t_1=t_2\\ \sigma^2p(t_1)p^*(t_2)\left(\dfrac{1-e^{\mathrm{j}2\pi t_{1N/T}}}{1-e^{\mathrm{j}2\pi t_{1/T}}}\right)\left(\dfrac{1-e^{\mathrm{j}2\pi t_{2N/T}}}{1-e^{\mathrm{j}2\pi t_{2/T}}}\right), & t_1\neq t_2\end{cases} \quad (式\ 13\text{-}20)$$

其中 $E[|a_n|^2]=\sigma^2$。从式 13-20 可看出，在采样点 kT_s $(k\in Z)$ 上，互相关函数的值永远为零，因此 OFDM 符号内的 N 个采样值为独立同分布的高斯随机变量，这也是从采样值相关性角度出发解释高 PAPR 出现的原因。采用相同的成形脉冲对各个子载波进行脉冲成形不会影响采样值之间的这种互相关特性，只会增加或保持传输信号的峰值幅度，使 PAPR 增大或保持不变。

定理 13.1：N 个子载波的 OFDM 系统，若每个子载波采用相同的成形脉冲，即 $p_n(t)=p(t)$ $(n=0,1,\cdots,N-1)$，则 OFDM 信号 PAPR 的最大值满足：

$$\mathrm{PAPR}_{\max}\geqslant N \quad (式\ 13\text{-}21)$$

上式当且仅当矩形脉冲时取等号。

证明：若采用相同成形脉冲，式 13-18 的最大值为：

$$\mathrm{PAPR}_{\max}=\frac{1}{N}\max_{0\leqslant t\leqslant T}\left(\sum_{n=0}^{N-1}|p_n(t)|\right)^2=N\max_{0\leqslant t\leqslant T}|p(t)|^2 \quad (式\ 13\text{-}22)$$

由等能量的条件，有下列不等式：

$$\int_0^T|p(t)|^2\,dt=T\leqslant\max_{0\leqslant t\leqslant T}|p(t)|^2T \quad (式\ 13\text{-}23)$$

将上式代入式 13-22，定理 13.1 得证。

从式 13-23 可以看出，只要成形脉冲满足 $\max\limits_{0\leqslant t\leqslant T}|p(t)|^2$，PAPR 最大值就达到其下界 N，所以明显地，定理 13.1 给出的当且仅当采用矩形脉冲时能达到这个抑制下界的阐述是局限的，应该说矩形脉冲只是其中的一类。

- 不同成形脉冲

若每个子载波采用不同的成形脉冲，那么式 13-19 可写为：

$$R_s(t_1,t_2)=\sigma^2\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}p_n(t_1)p_m{}^*(t_2)\exp[\mathrm{j}2\pi(nt_1-mt_2)/\mathrm{T}] \quad (式\ 13\text{-}24)$$

此时的互相关函数在采样点上的值完全由各子载波上的成形脉冲波形决定，因此适当地选择成形脉冲将增大OFDM符号各采样点之间的互相关值，从而达到抑制PAPR的目的。

定理 13.2：N 个子载波的 OFDM 系统，若每个子载波采用一组不同的成形脉冲，即 $\{p_0(t),p_1(t),\cdots,p_{N-1}(t)\}$，且有：

$$p_n(t)=\begin{cases} w(t-nT_s), & 0\leqslant t\leqslant T=NT_s \\ 0, & \text{other} \end{cases} \qquad (式\ 13\text{-}25)$$

其中 $w(t)$ 为周期和能量为 T 的周期信号，即有：

$$\int_0^T |w(t)|^2 \,\mathrm{d}t = T \qquad (式\ 13\text{-}26)$$

则 OFDM 信号 PAPR 最大值满足：

$$\mathrm{PAPR}_{\max} \leqslant N \qquad (式\ 13\text{-}27)$$

上式当且仅当矩形脉冲时取等号。

证明： 若采用不同的成形脉冲，式 13-18 的最大值为：

$$\mathrm{PAPR}_{\max}=\frac{1}{N}\max_{0\leqslant t\leqslant T}\left(\sum_{n=0}^{N-1}|w(t-nT_s)|\right)^2=\frac{1}{N}\left(\sum_{n=0}^{N-1}|w(nT_s)|\right)^2 \qquad (式\ 13\text{-}28)$$

对于较大子载波数 N，有：

$$\sum_{n=0}^{N-1}|w(nT_s)|=\frac{N}{T}\int_0^T|w(t)|\,\mathrm{d}t \qquad (式\ 13\text{-}29)$$

则式 13-28 变为：

$$\mathrm{PAPR}_{\max}=\frac{N}{T^2}\left(\int_0^T|w(t)|\,\mathrm{d}t\right)^2 \qquad (式\ 13\text{-}30)$$

利用施瓦茨不等式，可得：

$$\left(\int_0^T|w(t)|\,\mathrm{d}t\right)^2\leqslant T\int_0^T|w(t)|^2\,\mathrm{d}t=T^2 \qquad (式\ 13\text{-}31)$$

将上式代入式 13-30，定理 13.2 得证。

从式 13-31 可以看出，只有采用一组矩形脉冲时，OFDM 信号的 PAPR 最大值达到上界，这就是一般的 OFDM 系统。只要采用一组其他的不同成形脉冲对各个子载波进行脉冲成形就会降低传输信号的峰值幅度，使 PAPR 减小。

3. Nyquist 脉冲成形

上面分析和证明为 PS 技术抑制 PAPR 提供了理论基础。下面就是如何构造有效的成形脉冲集合。

首先必须明确集合内的成形脉冲都要满足前文中提到的四个条件，然后根据思想：将一个主脉冲通过循环移位组成的成形脉冲集合能使各子载波峰值不在同一时刻出现，最常见的可以使用 Nyquist 脉冲，此处定义按下列条件组成的 Nyquist 脉冲集合：

$$p_m(t)\mathrm{e}^{\mathrm{j}2\pi\frac{m}{T}t}=p_n(t-\tau_{m-n})\mathrm{e}^{\mathrm{j}2\pi\frac{n}{T}(t-\tau_{m-n})},\quad n,m=0,1,\cdots,N-1 \qquad (式\ 13\text{-}32)$$

其中 $\tau_{m-n}=[(m-n) \bmod N]T_s$，$p_n(t)$ $(n=0,1,\cdots,N-1)$ 为 Nyquist 脉冲，具有 ISI 性质：

$$p_n(kT_s)=\begin{cases}1, & k=0\\ 0, & k\neq 0\end{cases},\ k\in \mathrm{Z} \qquad (式\ 13\text{-}33)$$

由条件式 13-32 定义的 Nyquist 脉冲集合对应的 OFDM 信号 PAPR 最大值为：

$$\mathrm{PAPR}_{\max}=\frac{1}{N}\max_{0\leqslant t\leqslant T}\left(\sum_{n=0}^{N-1}\left|p_n(t)\right|\right)^2\leqslant\frac{1}{N}\left(\sum_{n=0}^{N-1}\max_{0\leqslant t\leqslant T}\left|p_n(t)\right|\right)^2=N \qquad (式\ 13\text{-}34)$$

当且仅当矩形脉冲时 PAPR 最大值为 N。上式的推导是利用了 Nyquist 脉冲的无 ISI 性质（式 13-33）。这个结论与定理 13.2 也是相符合的，而且表明所有按上述方式构造的 Nyquist 脉冲集合都能用于 OFDM 信号的 PAPR 抑制。

由于成形脉冲 $p_n(t)$ $(n=0,1,\cdots,N-1)$ 都是符号周期 T 内的时限信号，所以可用 Fourier 级数近似，即：

$$p_n(t)\approx\sum_{l=-L}^{N+L-1}c_{n,l}\mathrm{e}^{\mathrm{j}2\pi\frac{l}{T}t},\quad 0\leqslant t\leqslant T \qquad (式\ 13\text{-}35)$$

其中 $L=\lfloor N\beta/2\rfloor$，$c_{n,l}$ 为 $p_n(t)$ 的 Fourier 级数的系数：

$$c_{n,l}=\frac{1}{T}\int_0^T p_n(t)\mathrm{e}^{-\mathrm{j}2\pi\frac{l}{T}t}\mathrm{d}t=\frac{1}{T}P_n\left(\frac{l}{T}\right) \qquad (式\ 13\text{-}36)$$

将式 13-35 代入式 13-32，可得：

$$p_n(t)=\sum_{l=-L}^{N+L-1}c_{n,l}e^{-\mathrm{j}2\pi\frac{nl}{N}}e^{\mathrm{j}2\pi\frac{l-n}{T}t} \qquad (式\ 13\text{-}37)$$

将上式表达的各子载波波形代入式 13-17，得：

$$\begin{aligned}s(t)&=\sum_{n=0}^{N-1}a_n p_n(t)\mathrm{e}^{\mathrm{j}2\pi\frac{n}{T}t}\\&=\sum_{n=0}^{N-1}a_n\sum_{l=-L}^{N+L-1}c_{n,l}\mathrm{e}^{-\mathrm{j}2\pi\frac{nl}{N}}\mathrm{e}^{\mathrm{j}2\pi\frac{l-n}{T}t}\mathrm{e}^{\mathrm{j}2\pi n\frac{t}{T}}\\&=\sum_{n=0}^{N-1}a_n\sum_{l=-L}^{N+L-1}c_{n,l}\mathrm{e}^{-\mathrm{j}2\pi\frac{nl}{N}}\mathrm{e}^{\mathrm{j}2\pi\frac{l}{T}t}\\&=\sum_{l=-L}^{N+L-1}\left[\sum_{n=0}^{N-1}a_n c_{n,l}\mathrm{e}^{-\mathrm{j}2\pi\frac{nl}{N}}\right]\mathrm{e}^{\mathrm{j}2\pi\frac{l}{T}t}\\&=\sum_{l=-L}^{N+L-1}b_l\,\mathrm{e}^{\mathrm{j}2\pi\frac{l}{T}t}=\mathrm{IFFT(b)}\end{aligned} \qquad (式\ 13\text{-}38)$$

其中 $b_l=\sum_{n=0}^{N-1}a_n c_{n,l}\mathrm{e}^{-\mathrm{j}2\pi\frac{nl}{N}}$，$b=\{b_l\}$ 为包含 $N+2L$ 个元素的向量。令 $p_{n,l}=c_{n,l}\mathrm{e}^{-\mathrm{j}2\pi\frac{nl}{N}}$ $(n=0,1,\cdots,N-1,l=-L,\cdots,\cdots N+L-1)$，则 $P=(p_{n,l})$ 代表 $N\times(N+2L)$ 的正交矩阵，称为成形矩阵，$b=aP$ 为变换后的新序列。

常见的 Nyquist 脉冲有升余弦脉冲，本文在此基础上又采用了两种改进的 Nyquist 脉冲设计了成形脉冲集合应用于 OFDM 信号的 PAPR 抑制。

- 升余弦脉冲

升余弦脉冲的频率响应和时域信号分别如式 13-39 和式 13-40 所示：

$$P_1(f)=\begin{cases}1, & |f|\leqslant B(1-\alpha)\\ \dfrac{1}{2}\left\{1+\cos\left[\dfrac{\pi}{2\alpha B}\left(|f|-B(1-\alpha)\right)\right]\right\}, & B(1-\alpha)<|f|<B(1+\alpha)\\ 0, & |f|\geqslant B(1+\alpha)\end{cases} \quad (式\ 13\text{-}39)$$

$$p_1(t)=\sin c\left(\frac{t}{T_s}\right)\frac{\cos(2\pi\alpha t/T_s)}{1-4\alpha^2t^2/T_s^2} \quad (式\ 13\text{-}40)$$

其中，α $(0\leqslant\alpha\leqslant1)$ 为滚降系数。

- 改进 Nyquist 脉冲

改进 Nyquist 脉冲的频率响应和时域信号分别如式 13-41 和式 13-42 所示：

$$P_2(f)=\begin{cases}1, & |f|\leqslant B(1-\alpha)\\ e^{\lambda\left[B(1-\alpha)-|f|\right]}, & B(1-\alpha)<|f|\leqslant B\\ 1-e^{\lambda\left[|f|-B(1+\alpha)\right]}, & B<|f|<B(1+\alpha)\\ 0, & |f|\geqslant B(1+\alpha)\end{cases} \quad (式\ 13\text{-}41)$$

$$p_2(t)=\frac{1}{T_s}\sin c\left(\frac{t}{T_s}\right)\frac{4\lambda\pi t\sin(\pi\alpha t/T_s)+2\lambda^2\cos(\pi\alpha t/T_s)-\lambda^2}{(2\pi t)^2+\lambda^2} \quad (式\ 13\text{-}42)$$

其中参数 $\lambda=\dfrac{\ln 2}{\alpha B}$。

上面两种 Nyquist 脉冲都是实的对称信号，且在 Nyquist 采样频率处为零，具有无 ISI 性质。虽然由式 13-42 可以看出改进脉冲的时域波形拖尾是渐近 t^{-2} 衰减的，比升余弦脉冲渐进 t^{-3} 衰减得慢，但是它的旁瓣幅度比升余弦脉冲要小，也就是不同采样时刻叠加起来对其他值的干扰要少。

4．仿真及结果分析

图 13-9 是在不同调制方式下基于改进的 Nyquist 脉冲成形的 OFDM 信号 PAPR 的 CCDF 仿真曲线。其中，OFDM 信号分别采用 64QAM、16QAM、QPSK 和 BPSK 调制方式，子载波数 N=128，Nyquist 脉冲的滚降系数为 $\alpha=0.3$。从图 13-9 中可看出，在一般情况下，随着调制阶数的提高，OFDM 信号的 PAPR 也会相应地提高，QPSK 调制方式相比高阶调制方式对 OFDM 信号的 PAPR 有 2dB 以上的改善，但是，由于 Nyquist 脉冲的频谱

分布采用复数形式，QPSK 调制方式比低阶的 BPSK 调制方式具有更好的 PAPR 抑制性能，所以，在本系统的设计中采用了 QPSK 调制方式。

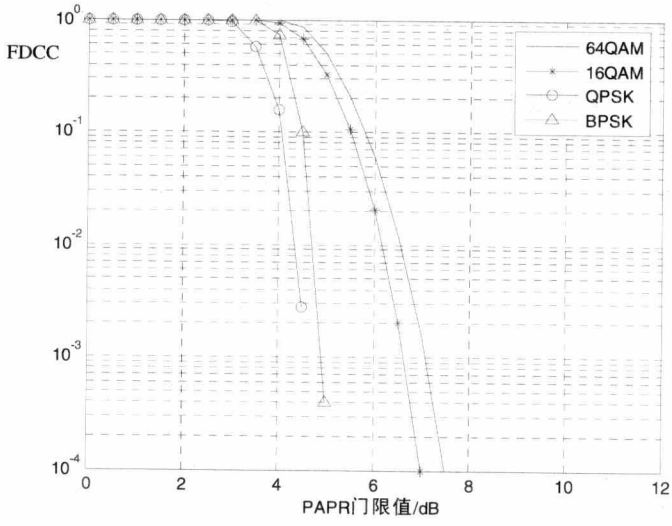


图 13-9　基于 Nyquist 脉冲整形的 OFDM 信号 PAPR 的 CCDF

图 13-10 是基于 Nyquist 脉冲成形的 OFDM 信号的 PAPR 的 CCDF 仿真曲线，为了了解这种方法的性能，此处还给出了原始信号和采用 8 路随机相位 SLM 的 OFDM 信号的 CCDF 分布进行比较。其中，OFDM 信号采用 QPSK 调制方式，子载波数 N=128，Nyquist 脉冲的滚降系数为 $\alpha = 0.3$。从图 13-10 中可以看出，Nyquist 脉冲成形技术对 OFDM 信号的 PAPR 特性有显著改善，比常用的 SLM 方法也有较大的提高；Nyquist 脉冲成形技术对信号 PAPR 的改善性能随脉冲类型的不同而变化，改进的 Nyquist 脉冲比常用的升余弦脉冲性能要好，原因在于改进的 Nyquist 脉冲具有较好的抗 ISI 特性，它的时域波形具有较小幅度的旁瓣，所以我们还可以寻找更好的 Nyquist 脉冲，进一步改善 OFDM 信号的 PAPR 分布。

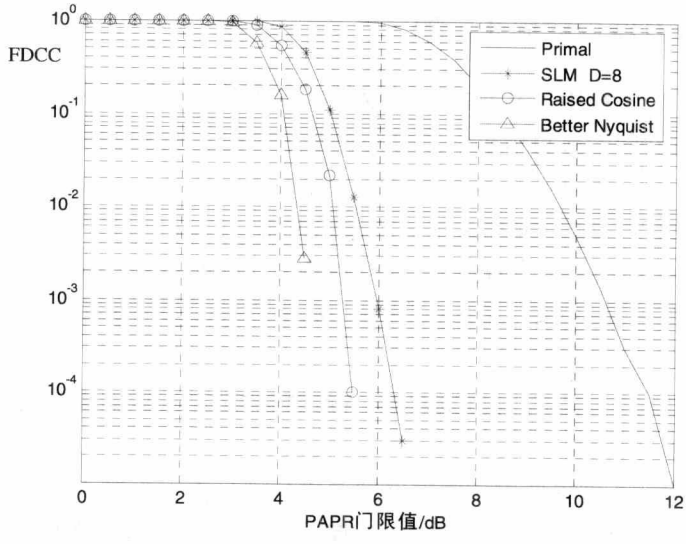


图 13-10　基于 Nyquist 脉冲整形的 OFDM 信号 PAPR 的 CCDF

图 13-11 为滚降系数 α 取不同值时，采用升余弦脉冲成形的 OFDM 信号 PAPR 的 CCDF 仿真曲线。图中实线对应原始信号 PAPR 的 CCDF。其中，OFDM 信号采用 QPSK 调制方式，子载波数 N=128，滚降系数 α 分别为 0.1、0.3 和 0.5。从图 13-11 中可以看出，Nyquist 脉冲成形技术对 OFDM 信号的 PAPR 的抑制随成形脉冲滚降系数的不同而变化，滚降系数越大，改善性能越好。这是因为滚降系数决定了成形脉冲时域信号的拖尾衰减速度，滚降系数越大，拖尾幅度衰减越快，成形脉冲抗 ISI 性能越强，这就说明 OFDM 各子载波在同一时刻出现的概率越小，所以成形脉冲对 PAPR 的抑制性能越好。

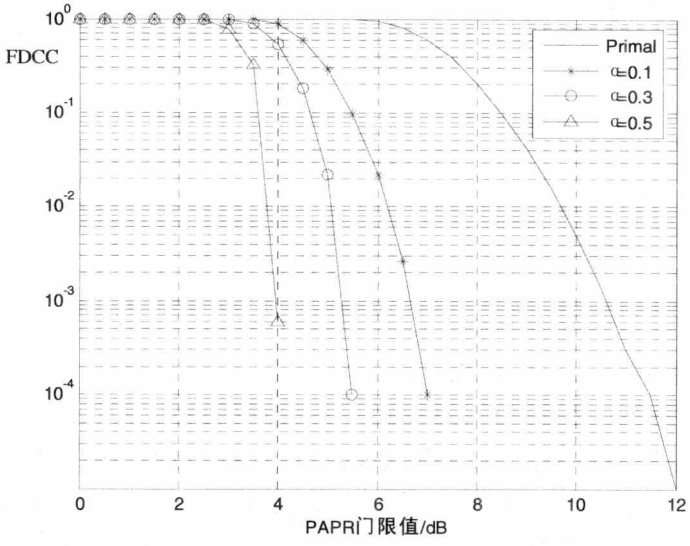


图 13-11　不同滚降系数的 OFDM 信号 PAPR 的 CCDF

13.3 OFDM 系统的同步算法设计

13.3.1　OFDM 系统中的同步问题

一般在 OFDM 系统中，从频域和时域两大方面考虑，同步问题可分为载波频率同步和时间同步，而时间同步又可以进一步分为符号定时同步和采样时钟同步。因此在 OFDM 系统中需要考虑三部分同步：符号定时同步、频率同步和采样时钟同步。

符号定时同步就是确定 OFDM 符号的起始位置，即每个 FFT 窗的位置。若符号同步的起始位置在循环前缀（CP）长度内，各个子载波之间的正交性依旧保持，而此时符号同步的偏差可以看作是由信道引入的相位旋转；如果符号同步的偏差超过了保护间隔，就会引入子载波间干扰（ICI）。

采样时钟同步用于在进行 A/D 转换时，确定接收端与发送端具有相同的采样时钟。采样时钟频率误差会引起 ICI。

频率同步是估计并校正数据流中存在的频率偏移。由于收发双方本振频率不匹配，加上多普勒效应导致接收信号的载波频率发生偏差，会使得子载波间正交性受到破坏。它对

码元的直接影响导致信号幅度衰落，并且给系统带来载波间干扰，严重影响系统的性能。

各种同步对于系统的影响是不一样的，有的仅仅使接收端信号产生一定的相位偏移，有的则会导致接收信号的采样点不是一个完整的 OFDM 符号信息，严重的会直接影响整个传输系统的性能。所以，需要根据系统的要求，设计相关的同步算法。

13.3.2　同步偏差对 OFDM 信号的影响

为了设计 OFDM 通信系统同步的方法，这里先简单分析频率偏差和定时偏差对 OFDM 信号造成的影响。

1．频率偏差对信号的影响

发送的数字信号可以表示为：

$$s_n = \sum_{i=0}^{N-1} d_i \cdot \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}in} \qquad n = 0,1,\cdots,N-1 \qquad \text{（式 13-43）}$$

假设信号通过加性的高斯白噪声信道，信道的离散时域和频域响应分别记为 h_n 和 H_n，并且接收符号同步理想。设接收机与发射机之间的频率差为 ΔF，定义频偏系数为 $\alpha = \Delta F/(1/T_{FFT})$。则接收机接到的信号可以表示为：

$$r_n = (s_n * h_n)\cdot \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}\alpha n} = \frac{1}{N_s}\sum_{i=0}^{N_s-1} d_i H_i \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(i+\alpha)n} + w_n \qquad \text{（式 13-44）}$$

经过解调，即 FFT 变换得到：

$$\begin{aligned} R_i &= \sum_{n=0}^{N-1} r_n \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}ni} = d_i \cdot \left(\frac{H_i}{N_s}\sum_{n=0}^{N-1}\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}\alpha n}\right) + \frac{1}{N_s}\sum_{n=0}^{N-1}\sum_{\substack{m=0\\ m\neq i}}^{N_s-1} d_m H_m \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(m+\alpha-1)n} + W_i \\ &= d_i\lambda_i + I_i + W_i \qquad i = 0,1,\cdots,N_s-1 \end{aligned} \qquad \text{（式 13-45）}$$

由 Pollet 的推导可知，要使 OFDM 系统可以正常通信，并且频偏造成的信噪比损失较小（小于 0.1dB），频率同步的要求是使频偏系数 α 小于 1%。

2．定时偏移对信号的影响

定时的偏移会引起子载波相位的旋转，而且相位旋转角度与子载波的频率有关，频率越高，旋转角度越大，这可由傅里叶变换的性质来解释：时域的频偏对应于频域的相位旋转。设解调的 FFT 窗口的起始位置为 $n = n_0 - \Delta n$，其中 Δn 表示定时偏移。则 Δn 造成的影响可以用数学模型表示为：

$$R_k = d_k \mathrm{e}^{\mathrm{j}2\pi k\Delta n} \qquad \text{（式 13-46）}$$

如果定时的偏移量与最大时延扩展的长度之和仍小于循环前缀的长度，此时子载波的正交性仍然成立，没有 ISI 和 ICI，对解调出来的数据信息符号的影响只是一个相位旋转。

13.3.3 OFDM 同步算法概述

现有的关于 OFDM 同步的算法从利用数据方面而言，主要沿袭下面两条思路：

- 数据辅助型，即基于导频符号，这类算法的优点是捕获快、精度高，适合分组数据通信，具体的实现是在分组数据包的包头加一个专门用来做定时、频偏估计的 OFDM 训练符号。
- 非数据辅助型，即盲估计，它利用 OFDM 信号的结构，例如，由于加循环前缀使 OFDM 的前端与后端有一定的相关性、利用虚子载波来做估计以及利用数据经过成形滤波之后的循环平稳性等方法来做估计。

基于训练符号的同步算法是在时域上将已知信息加入待发 OFDM 符号。通常置于 OFDM 符号前或由多个 OFDM 符号构成的帧的前部。训练符号的加入可以同时完成同步和信道估计。而对基于训练符号的同步算法的研究主要是两个方面：训练符号的结构组成和训练符号的码型。

OFDM 信号的同步也可以充分利用信号本身的特点展开，即所谓的非数据辅助型同步算法就是基于这种思路。由于 OFDM 符号之间存在循环前缀 CP，考查相隔为 N 的两个接收样本点之间的相关性。如果这两个样本点中一个属于前缀，一个属于同一个 OFDM 码元之内的复制信息，则两者的相关性大；如果一个属于 CP，一个属于不相关信息，则两者的相关性较小。基于 CP 的同步算法正是这样的思想。实际中典型的算法是最大似然估计算法（Maximum Likelihood，ML）。

对于数据辅助型的同步算法而言，频偏估计是通过使用导频符号或训练序列得到的，系统传输效率因而受到了损失，但其估计精度要比非数据辅助型同步算法高很多。

一个典型的 OFDM 系统的同步实现框图如图 13-12 所示。

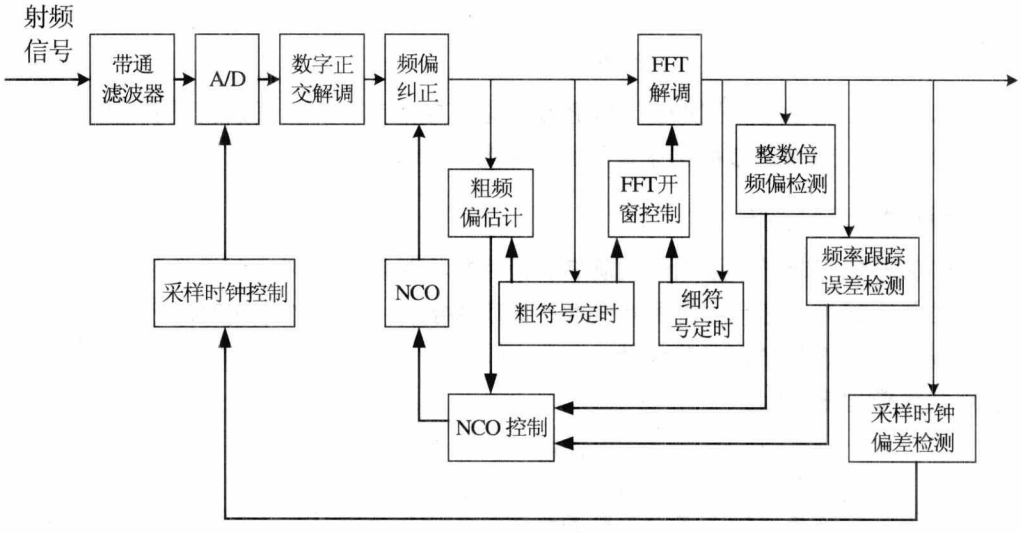


图 13-12　OFDM 系统同步实现的典型框图

首先利用训练序列或者插在数据前面的保护间隔做同步定时粗估计，得到时域信号的同步头的位置，同时粗略估计出频率偏差。通过定时/频偏控制单元校正接收到的数据，同时用估计到的同步起始位置定出 FFT 数据处理窗口。数据经过 FFT 变换后，首先利用数据内插的频域导频（Pilot）做整数倍频偏估计，然后利用导频的相位变化信息估计出小数倍子载波间隔的细频偏，并将这两个估计出的频偏送到定时/频偏控制单元和先前估计出的粗频偏一起去频偏校正单元做数据频偏校正。利用数据间内插的频域导频，还可以估计出用于定时同步估计偏差引起的数据相位偏转、公共相位误差及 A/D 采样钟偏移，将估计的值分别送到 FFT 开窗控制单元、相位校正单元和晶振单元去校正相应的误差。

13.3.4　OFDM 系统的同步设计

基于 OFDM 技术通信系统经常应用于突发数据业务，一般需要在很短的时间内捕获时偏和频偏，要求在解调各个子载波之前去除 ICI，这就需要利用 FFT 之前的训练序列来达到快速同步。在突发传输系统中，系统子载波数比较少，传输的数据帧比较短，可以不用考虑采样时钟频率误差对系统的性能影响，同时也不用考虑定时跟踪。因此在 OFDM 信号进行解调之前，必须至少先完成定时同步和载波同步。本系统采用的同步方案是利用时域的训练序列做定时、频偏的联合估计。

在设计合理的同步方案时需要考虑一些实际的问题：首先是时偏和频偏的相互影响，如定时的准确是以频率偏移已纠正为前提条件，频率偏移的估计算法又是以定时准确为前提等；第二，要使同步算法的性能与开销矛盾得到折中；最后是同步引导与系统的有效载荷矛盾，由于增加同步引导虽然可以使同步性能上升，但系统效率会下降。要尽量使得最少的同步引导达到最好的同步性能。

这里主要参考了一些已有的同步算法及协议，提出一种适用于突发传输机制的 OFDM 系统的同步方案，方案中包括帧结构的设计、各种同步算法、具体的同步流程。

1．帧结构设计

利用训练序列进行同步，首先要构造帧的前导结构。基本结构类似于 IEEE 802.11a 的帧的前导结构，但是由于传输速率的不同与同步流程的差异，在具体的结构上是有区别的。本系统考虑使用尽量少的同步帧头开销来达到系统同步的要求，采用的前导结构如图 13-13 所示。

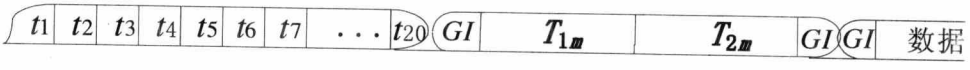


图 13-13　基于 OFDM 的通信系统的前导结构

从图 13-13 中可以看到前导序列的长度为两个 OFDM 符号，每个 OFDM 符号包括 320 个样值点，前面两帧主要用于同步实现。

图 13-13 中 t_1 到 t_{10} 是短训练符号，T_{1m} 和 T_{2m} 是各自的子载波个数为 128 的长训练符号。每个短训练符号由 16 个子载波组成，短训练序列是由伪随机序列经过数字调制后插 0，再经过 IFFT 得到的。具体过程如下：首先采用抽头系数为[1 0 0 1]的 4 级移位寄存器产生

长度为 15 的伪随机序列之后末尾补 0，经过 QPSK 调制后的伪随机序列只在 16 的整数倍位置上出现，其余的位置补 0，产生长度为 128 的序列，此序列再补 128 个 0 经过数据搬移后做 256 点的 IFFT 变换就得到 16 个以 16 为循环的训练序列，实虚部分别如图 13-14 及图 13-15 所示，经过加循环前后缀就会产生 20 个相同的短训练序列。

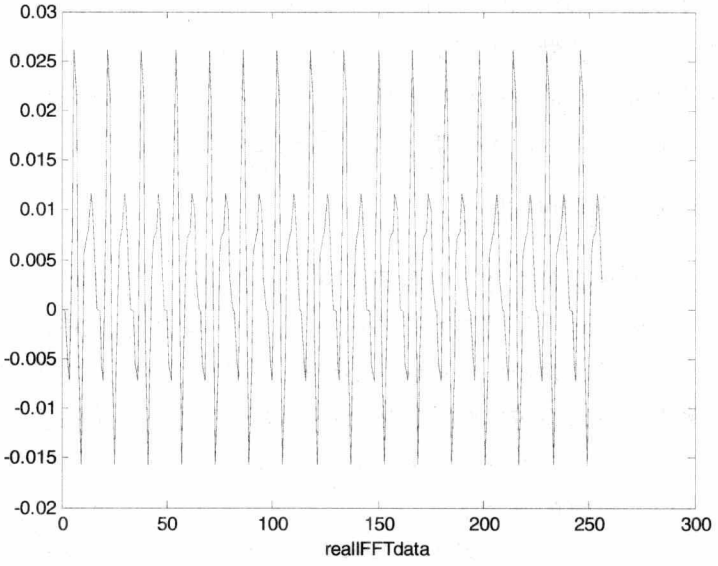


图 13-14　短训练序列实部波形

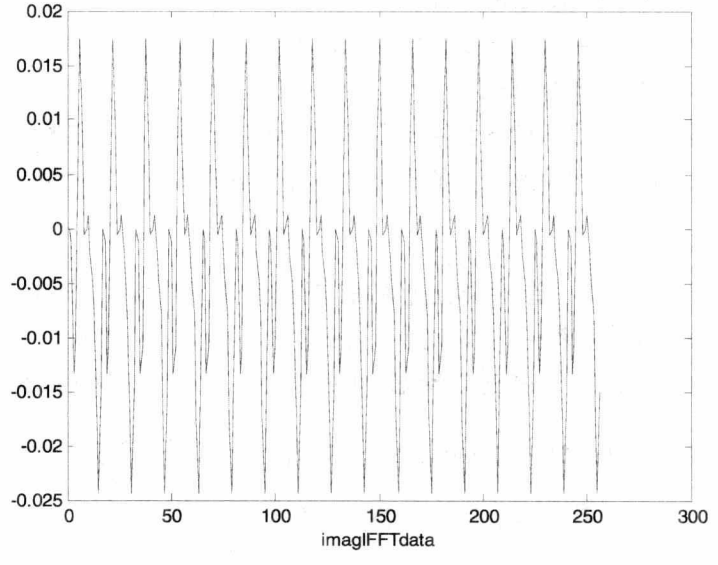


图 13-15　短训练序列虚部波形

T_{1m} 和 T_{2m} 的产生与短训练序列过程基本一样，使用的是抽头系数为[1 0 0 0 0 0 1]的 7 级移位寄存器产生长度为 127 的伪随机序列，然后末尾补 0，经过数字调制之后的序列只出现在 2 的整数倍位置上，其后的过程与短训练序列方式一样，其 IFFT 输出的实虚部波

形分别如图 13-16 及图 13-17 所示。

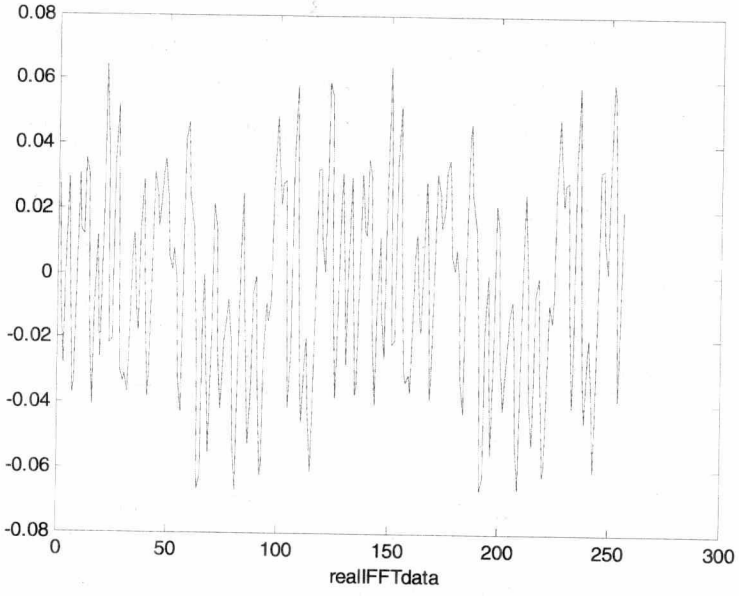


图 13-16　T_{1m} 和 T_{2m} 的实部波形

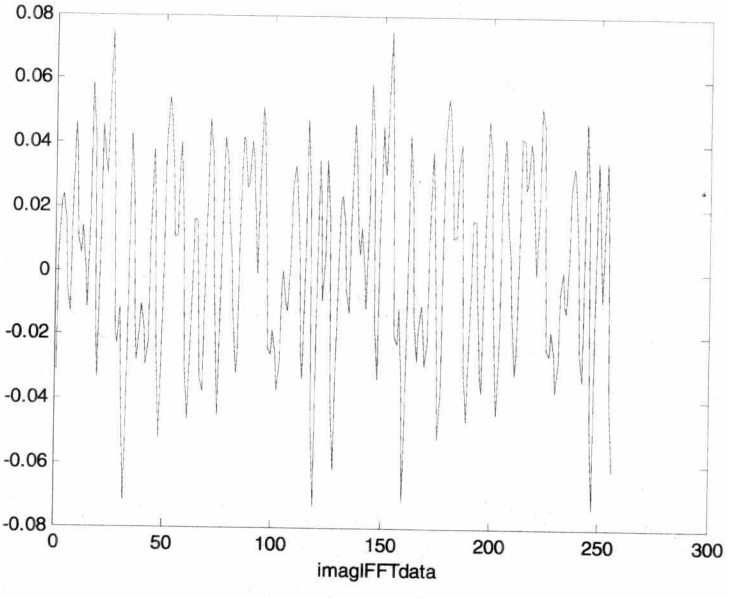


图 13-17　T_{1m} 和 T_{2m} 的虚部波形

系统中十个短训练符号主要完成定时粗同步和频率粗估计，而 T_{1m} 和 T_{2m} 主要完成定时细同步与频率细估计。

2．定时同步算法及仿真

基于突发数据业务的定时同步主要包括定时粗同步和定时细同步，其中粗同步又称为帧捕

获，用于检测帧的到来，定时细同步是检测帧头的精确位置，以确定解调端 FFT 的开窗位置。

- 定时粗同步

利用前导结构中短训练序列周期性做定时粗同步最典型的算法就是延时自相关算法，其实现框图如图 13-18 所示，其中滑动窗口 C 计算接收信号和接收延时 D 个采样点的互相关系数，延时 D 等于短训练符号周期；滑动窗口 P 计算接收信号的能量。

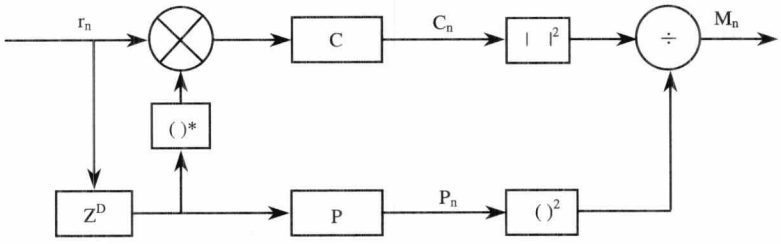


图 13-18　延时自相关法的框图

图 13-18 中的参数 c_n 和 p_n 分别通过式 13-47 与式 13-48 计算得到：

$$c_n = \sum_{k=0}^{L-1} r_{n+k} r^*_{n+k+D} \qquad (式\ 13\text{-}47)$$

$$p_n = \sum_{k=0}^{L-1} r_{n+k+D} r^*_{n+k+D} = \sum_{k=0}^{L-1} \left| r_{n+k+D} \right|^2 \qquad (式\ 13\text{-}48)$$

其中，L 为短训练符号的长度（$L=D$）。

帧检测的判决函数为：

$$m_n = \frac{\left| c_n \right|^2}{\left(p_n \right)^2} \qquad (式\ 13\text{-}49)$$

由于经过了归一化处理，上述判决函数的大小与接收信号功率无关。在工程实现的时候，c_n 和 p_n 的计算可以通过移动递归求和来实现，以降低计算的复杂度和硬件资源开销。递归求和的计算公式如下：

$$c_{n+1} = c_n + r_{n+L} r^*_{n+2L} - r_n r^*_{n+L} \qquad (式\ 13\text{-}50)$$

开始阶段，由于没有进行频率同步，收发端载波频率偏差会比较大，但是通过式 13-47 与式 13-48 可见，自相关算法可以很好地克服较大频率偏差带来的影响，适合起始阶段的帧捕获。

图 13-19 为高斯白噪声信道大信噪比下延时自相关帧检测的仿真结果图，可以看出当数据开始的附近判决函数迅速跳变为最大值，并保持一个平台期。

在信噪比较低的情况下，判决函数的平台将发生一定的抖动，如图 13-20 所示。

从图 13-20 中可以看出，即使在 SNR=0 的情况下，判决的平台仍能保持得较好。

可以通过设定合适的门限，检测有连续 M 个样值点超过门限值时就判决训练帧位置的出现。并通过短训练序列的周期性，对较大范围的频率偏差进行调整，使得频率偏差限定在较小的范围内，在第 13.3.4 节中会详细介绍。

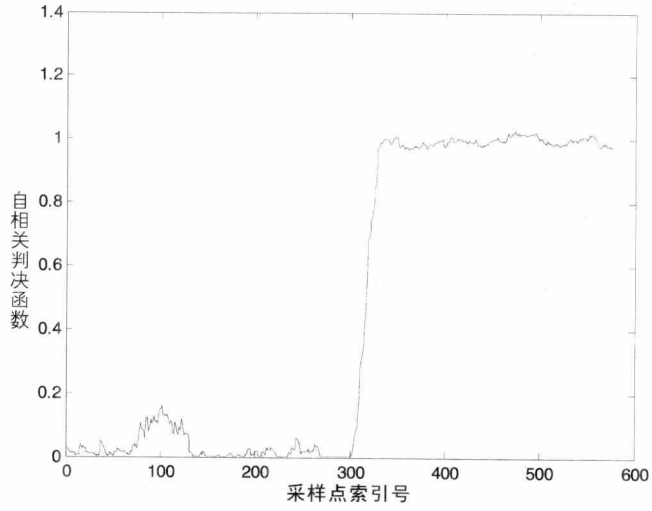


图 13-19　高斯白噪声信道下（SNR= 20dB）延时相关检测的仿真图

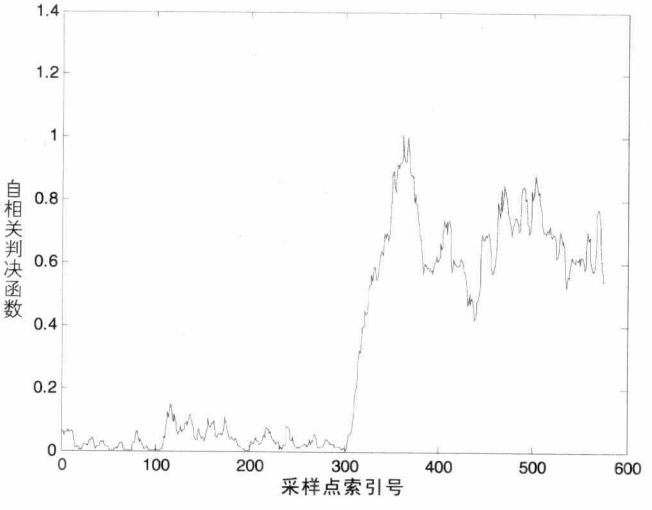


图 13-20　高斯白噪声信道下（SNR= 0dB）延时相关检测的仿真图

• 定时细同步

通过短训练帧的帧捕获，可以知道帧出现的大概位置，但是不能确定精确位置，为了实现定时的细同步，长训练帧的 T_{1m} 和 T_{2m} 与本地存储的训练符号进行相关运算，以确定 FFT 的窗口位置。本地存储的长训练符号为 $\{s\_long_i; i=0,1,\cdots,L-1\}$（$L$ 为本地存储的 T_{1m} 和 T_{2m} 符号的长度），互相关检测的判决函数为：

$$m_n = \left| \sum_{k=0}^{L-1} r_{k+n} \bullet s\_long_k^* \right| \qquad （式 13\text{-}51）$$

图 13-21 为本地互相关的仿真图。

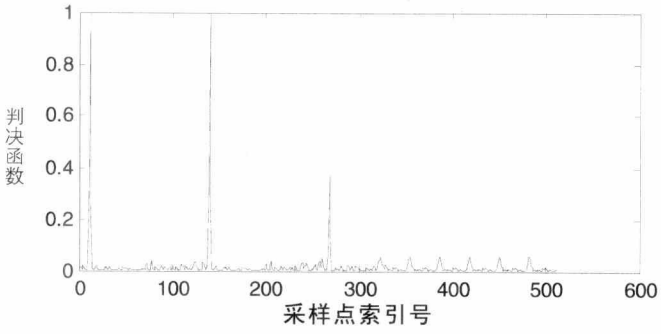


图 13-21　高斯白噪声信道大信噪比下本地互相关检测的仿真图

由于信道衰减和 AGC 的影响，互相关检测的峰值产生波动，一种解决的办法是先对接收信号进行量化，然后再进行相关检测，量化器如式 13-53 所示。

$$\Theta(x) = sign(\mathrm{Re}\{x\}) + \mathrm{j} \cdot sign(\mathrm{Im}\{x\}) \qquad (式\ 13\text{-}52)$$

式中：

$$sign(x) = \begin{cases} +1, & x \geqslant 0 \\ -1, & x < 0 \end{cases} \qquad (式\ 13\text{-}53)$$

经过量化的互相关检测仿真结果如图 13-22 所示，可以看出仿真结果与图 13-21 基本一致。量化操作也可以降低互相关检测算法在硬件实现时的复杂度和硬件资源开销。

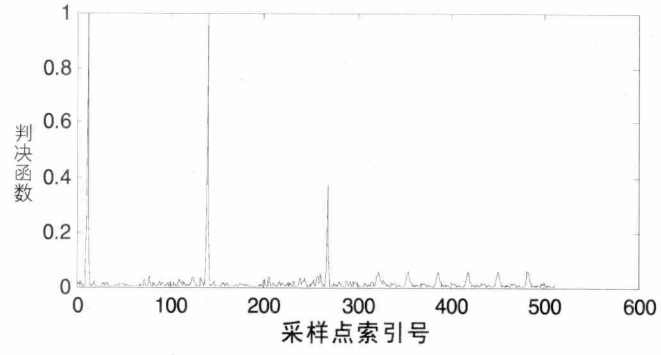


图 13-22　高斯白噪声信道大信噪比下量化后本地互相关检测的仿真图

在本系统中，我们将第一个过门限值的本地互相关峰值检测位置定为精确定时点，并以此实现 FFT 的开窗位置。

通过前导结构的两个训练帧的延时自相关算法和本地互相关检测，可以实现精确度非常高的定时同步。

3. 载波同步算法及仿真

发送端和接收端振荡器振荡频率的不匹配性或因为热漂移产生振荡频率的抖动，导致接收中频信号或基带信号的相位与对应的发送信号之间存在相位差，这种相位差与频率差有关。频率偏差越大，相位差也越大。而且这种频差对相差的影响具有累加性，前一个符

号的相位差会直接传给后一个符号。本节主要介绍设计系统所采用的频率同步算法及相位补偿算法。

- 频率同步

根据第 13.3.3 节的介绍，频率同步可以粗略地分为数据辅助和非数据辅助两大类。在本系统中，使用的是数据辅助的算法。

此处训练序列至少需要包括两个重复的符号。当发送时域数据信号 x_k 时，重复的通频带等效信号为：

$$s_k = x_k \mathrm{e}^{\mathrm{j}2\pi f_{tx} k T_S} \tag{式 13-54}$$

其中，f_{tx} 为发送载波频率。在接收端忽略噪声的情况下，重复的基带等效信号为：

$$r_k = x_k \mathrm{e}^{\mathrm{j}2\pi f_{tx} k T_S} \mathrm{e}^{-\mathrm{j}2\pi f_{rx} k T_S} = x_k \mathrm{e}^{\mathrm{j}2\pi \Delta f k T_S} = x_k \mathrm{e}^{\mathrm{j}\frac{2\pi\varepsilon k}{N}} \tag{式 13-55}$$

其中，f_{rx} 为接收载波频率，$\varepsilon = N\Delta f T_s = N(f_{tx} - f_{rx})T_s$ 为归一化载波频率偏差。定义两个连续重复符号之间的延时为 D 个采样点，符号长度为 L，定义中间变量：

$$R = \sum_{k=0}^{L-1} r_k r^*_{k+D} = \mathrm{e}^{-\mathrm{j}\frac{2\pi D\varepsilon}{N}} \sum_{k=0}^{L-1} x_k x^*_{k+D} = \mathrm{e}^{-\mathrm{j}\frac{2\pi D\varepsilon}{N}} \sum_{k=0}^{L-1} |x_k|^2 \tag{式 13-56}$$

从而得到归一化载波频率偏差的估值为：

$$\hat{\varepsilon} = -\frac{N}{2\pi D} \angle R \tag{式 13-57}$$

频偏估计范围为 $|\varepsilon| < N/2D$。

由式 13-57 可知，频偏的估计范围由 N、D 来决定。在本系统中，短训练序列中 $N=320$、$D=16$、$L=128$，因此频偏估计范围为 $|\varepsilon|<10$（2000Hz）；若取长训练序列，则 $N=320$、$D=128$、$L=128$，频偏估计范围为 250Hz。前一个方法有较大的纠偏范围，但是估计值得到的方差较大；后一个方法的纠偏范围较小，但是得到的方差较小。因此，可以将上述两种方法相结合，应用前导序列的短训练符号作频率粗估计，应用长训练符号作频率细估计，频率跟踪可以用循环前后缀的周期重复性并使用上述方法来完成，其中 $D=256$、$L=48$，频偏估计范围为 125Hz。这样就完成了系统的频率同步。

本系统的频率同步实现结构如图 13-23 所示。

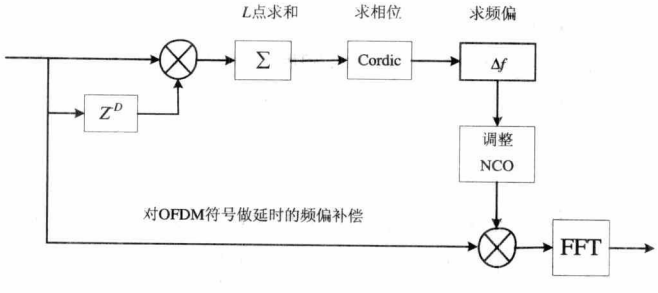


图 13-23 频率同步算法实现结构

频率粗同步、频率细同步及频率跟踪对应的 D、L 是不同的，但实际的 ASIC 设计中可以采用流水线的方式，所以乘法器和累加器只需要一套。可以使用数控震荡器（NCO）来纠正频率偏移，NCO 采用 CORDIC 算法来实现，这样可以简化硬件的复杂程度。

- 相位补偿

图 13-23 这个锁频环路只能纠正频率的偏差，不能纠正相位的偏差，图 13-24 给出了本系统所设计的 OFDM 系统在 QPSK 调制方式下且载波频率误差在 0.5%以下时，由于相位误差星座图发生偏转的情况。

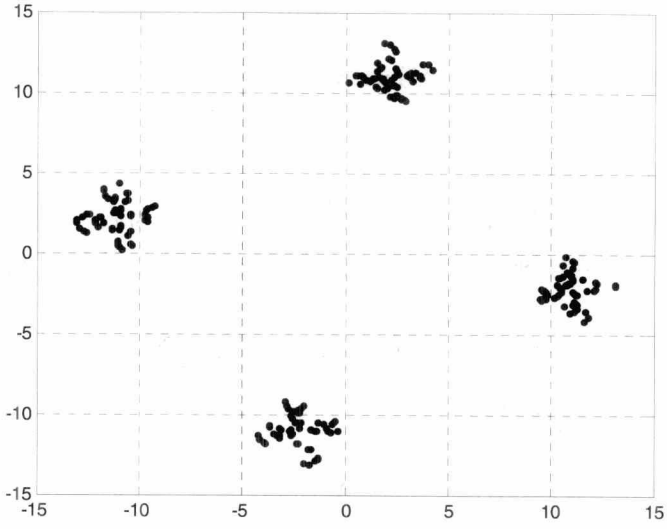


图 13-24　OFDM 系统在 QPSK 调制下未经相位补偿的接收端星座图

在频域系统需要导频所提供的信息来做相位补偿，进而消除相位偏差对系统的影响，若记相应的相位偏差为 θ，那么经过 FFT 之后可得：

$$Y_{i,k} = P_{i,k} H_{i,k} \mathrm{e}^{\mathrm{j}\theta} \qquad (式\ 13\text{-}58)$$

其中 i 为 OFDM 的符号数，k 为第 i 个符号的第 k 个数据，$P_{i,k}$ 为导频，$H_{i,k}$ 为信道冲击响应。假设接收信道估计结果完全正确，定义中间变量：

$$R = \sum_{k \in c} Y_{i,k} (P_{i,k} H_{i,k})^* = \mathrm{e}^{\mathrm{j}\theta} \sum_{k \in c} \left| P_{i,k} H_{i,k} \right|^2 \qquad (式\ 13\text{-}59)$$

则可以得到残余相位偏差的估计值为：$\hat{\theta} = \angle R$。

经过相位补偿后，从星座图上来看 QPSK 调制的数据已经基本上旋转回到星座图上 45 度的位置，如图 13-25 所示。

4．同步流程

本节设计的基于突发数据业务模式的 OFDM 通信系统采用的同步流程主要步骤如下：

- 定时粗同步。此步在 FFT 之前实现，为了实现帧头捕获。通过前导序列的短训练符号的自相关运算，由于不受初始频率偏差的影响，只要检测器的输出有连续 N 个值

（本系统 N 取值为 10）超过门限值，我们就判断帧头出现。

- 频率粗同步。短训练帧的自相关平台出现后，可以通过基于数据辅助类型的频率同步算法实现频率同步，应用短训练帧的重复性，可以估计的频偏范围在 2000Hz 以内，所以可以调整较大的频率偏差。

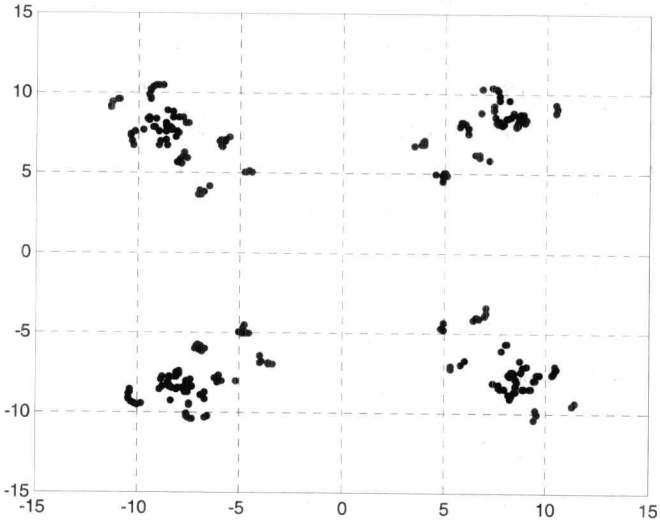


图 13-25　OFDM 系统在 QPSK 调制下经过相位补偿的接收端星座图

- 定时细同步。此步在 FFT 之前实现，为了实现帧头的精确定位。利用前导序列的长训练符号与本地存储训练符号的互相关特性，由于频率偏差已经调整到较小的范围，互相关运算后会出现两个明显的峰值。通过峰值位置可以确定 FFT 的精确开窗位置。
- 频率细同步。通过上面介绍的频率同步算法，应用短训练帧的重复性，可以估计的频偏范围在 250Hz 以内，所以可以调整频率偏差在较小的范围之内。
- 频率跟踪。同样应用上面的频率同步算法，利用每个 OFDM 符号的循环前后缀的重复特性，可以估计的频偏范围在 125Hz 以内，所以可以调整频率偏差在更精确的范围之内。
- 相位补偿。在频域系统中通过每个 OFDM 符号中的导频所提供的信息来做相位补偿，进而消除相位偏差对系统的影响，使 OFDM 的数据能正确解调。

13.4　OFDM 系统的编码算法设计

13.4.1　通信系统的信道编码

1. 信道编码的任务

当调制好的信号在信道里进行传输的时候，必然要受到信道的影响。信道的影响可以分成以下三个主要方面：第一是信道本身对信号产生的衰落：由于信道本身频率响应特性

不理想，造成对信号的破坏；第二是信道中的各种噪声，如背景噪声、脉冲噪声等，这些噪声叠加在信号上面，改变信号的幅度、相位和频率，使信号在解调时产生错误；第三是信号在传输过程中由于反射、折射或沿不同路径传播从而带来的叠加效应，即通常所说的多径效应，这会带来时间上前后信号互相干扰。总而言之，这三种影响都会导致在接收端信号解调的错误，使系统的误码率大大增加。

因此在一个实用的通信系统中，必须采取一定的措施来纠正错误，提高系统的误码率性能。信道编码就是一种非常有效的措施。信道编码的任务就是，在发送端以可控的方式在信号中加入一定的冗余度，而在接收端这些冗余度可以用来检测并且纠正信号通过信道后产生的错误。当然，冗余度的加入降低了系统的工作效率，但是和系统误码率的降低（即信号更加正确地传送）相比，这些代价是可以接受的。

2. 信道编码的原理

在一个噪声信道中，如果我们把调制/解调器和检测器看作是信道的一个组成部分。那么一个数字通信系统模型可以用图 13-26 表示。

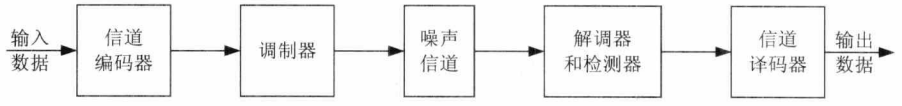


图 13-26　数字通信系统模型

信道编码器的作用是以可控的方式在二进制信息序列里插入一些冗余度，以便在接收端利用这些冗余度来克服信号在信道中受到的干扰和噪声影响。编码的一般过程是：每次取 k 个比特的信息序列，把这个 k 比特信息组映射成与之唯一对应的 n 比特组，这些 n 比特组称为码字。在这种方式中，由信道编码引入的冗余量可以用比值 n/k 来衡量，该比值的倒数，即 k/n 称为码率。

信道编码器输出的二进制序列被送入调制器，进入信道。调制器把二进制序列映射成可以在噪声信道里传播的波形。调制器可以把每个二进制比特映射成两个可能的波形之一：即 0 映射为波形 $s_1(t)$，1 映射为波形 $s_2(t)$；或者采用 $M=2^q$ 个可能的波形，每次把一个 q 比特组映射为其中一个可能的波形。

在接收端，解调器处理受损的接收波形，把波形映射为一个标量或矢量，作为发送的二元或 M 元数据符号的估值。然后检测器根据这个估值作判决输出送到信道译码器，利用可用的冗余度纠正信道和噪声产生的干扰。如果把检测器的判决过程看作是一种量化的形式，即把检测器看做是 Q 元的量化器。那么 $Q=2$，即对解调器输出作二值量化，称为硬判决；当 $Q>2$ 时，对解调器输出作 Q 值量化，称为软判决，当然这里要求 $Q \geqslant M$。

假设图 13-26 中信道编码器的输出为 q 元信号，即 $X=\{x_0,x_1,...,x_{q-1}\}$。检测器的输出为 Q 元信号，即 $Y=\{y_0,y_1,...,y_{Q-1}\}$，$Q \geqslant M=2^q$。

在 X 中任取 n 个元素，相应信道的输出为 Y 中相应的 n 个元素 $v_1,v_2,...,v_n$，如果联合条件概率：

$$P(Y_1=v_1,Y_2=v_2,...,Y_n=v_n \mid X_1=u_1,X_2=u_2,...,X_n=u_n)$$
$$=\prod_{k=1}^{n}P(Y_k=v_k \mid X_k=u_k) \quad (式\ 13\text{-}60)$$

那么图 13-26 中的假想信道就称为离散无记忆信道（Discrete Memoryless Channel，DMC）。这个信道的输入信号集合是 $X=\{x_0,x_1,...,x_{q-1}\}$，输出的信号集合是 $Y=\{y_0,y_1,...,y_{Q-1}\}$，信道的转移概率集合 $\{P(y_i \mid x_j)\}$ 如式 13-61 定义：

$$\{P(y_i \mid x_j)\}=\{P(Y=y_i \mid X=x_j)\} \quad (式\ 13\text{-}61)$$

假设发送的信号是 x_j，接受到的信号是 y_i，那么由事件 $Y=y_i$ 发生所提供的关于事件 $X=x_j$ 的互信息是 $\log[P(y_i \mid x_j)/P(y_i)]$。其中

$$P(y_i)=P(Y=y_i)=\sum_{k=0}^{q-1}P(x_k)P(y_i \mid x_k) \quad (式\ 13\text{-}62)$$

因此，输出 Y 为输入 X 提供的平均互信息量是

$$I(X;Y)=\sum_{j=0}^{q-1}\sum_{i=0}^{Q-1}P(x_j)P(y_i \mid x_j)\log\frac{P(y_i \mid x_j)}{P(y_i)} \quad (式\ 13\text{-}63)$$

其中，输入信号的发生概率 $P(x_j)$ 受到信道编码器的控制，而转移概率 $P(y_i \mid x_j)$ 是由信道的特征决定的。对于一组给定的输入信号概率 $P(x_j)$，$I(X;Y)$ 的最大值仅仅取决于转移概率 $P(y_i \mid x_j)$，即 DMC 信道的特性。这个最大值称为 DMC 信道的信道容量，用符号 C 表示：

$$C=\max_{P(x_j)}I(X;Y)$$
$$=\max_{P(x_j)}\sum_{j=0}^{q-1}\sum_{i=0}^{Q-1}P(x_j)P(y_i \mid x_j)\log\frac{P(y_i \mid x_j)}{P(y_i)} \quad (式\ 13\text{-}64)$$

这个 $I(X;Y)$ 值的最大化是在下述限制条件下进行的：

$$\begin{cases}P(x_j)\geqslant 0\\ \sum_{j=0}^{q-1}P(x_j)=1\end{cases} \quad (式\ 13\text{-}65)$$

式 13-63 中以 2 为底取对数，C 的单位是信道上每输送一个符号所能传递的比特数（bit/symbol）；如果以 e 为底取对数，单位是每输入符号传送的奈特数（nat/symbol）。如果每个符号间隔是 τ 秒，那么单位时间的信道容量为 C/τ，单位是比特/秒（bit/s）或奈特/秒（nat/s）。

一般的，求取 DMC 信道容量，使 $I(X;Y)$ 最大化，输入符号的发生概率集 $\{P(x_j)\}$ 必须满足的充要条件是：

$$\begin{cases} I(x_j;Y)=C & \text{对于所有满足}P(x_j)>0\text{的}j \\ I(x_j;Y)\leqslant C & \text{对于所有满足}P(x_j)=0\text{的}j \end{cases} \quad \text{（式 13-66）}$$

此处 C 为信道容量，且

$$I(x_j;Y)=\sum_{i=0}^{Q-1}P(y_i\mid x_j)\log\frac{P(y_i\mid x_j)}{P(y_i)} \quad \text{（式 13-67）}$$

通常，检验等概集是否满足条件比较容易，若不满足，则必须找出一个满足条件的非等概集 $\{P(x_j)\}$。

下面给出带限 AWGN 波形信道在平均功率受限的输入条件下单位时间的信道容量：

$$C=W\log\left(1+\frac{P_{av}}{WN_0}\right) \quad \text{（式 13-68）}$$

上述信道容量公式的意义在于确定噪声信道中可靠通信的传输速率上限。可靠通信的传输速率由 Shannon 在 1948 年提出的噪声信道编码定理给出。

噪声信道编码定理为：只要传输速率 R 小于信道容量 C，总存在一种信道编码及相应的解码器，以所要求的任意小的差错概率实现可靠通信；反之，如果 R 大于 C，则不可能有一种信道编码能使差错概率趋近于零。

3. 信道编码的分类

分组码（Block Coding）和卷积码（Convolutional Coding）是信道编码的两种主要形式。分组码的特点是一次处理相当规模的信息块（通常为成百字节的数据）；而卷积码则是处理串行的信息比特流。现在，人们已经设计出了许多种有用的分组码和卷积码，相应地也有许多种有效的译码算法，适用于不同的场合把原始信息从接收数据里面恢复出来。

分组码由一组固定长度称为码字的矢量构成。码字的长度就是矢量元素的个数，用 n 来表示。码字的元素选自有 q 个元素的字符集合。如果 q=2，则称为二进制分组码；如果 q>2，则称为非二进制分组码。长度为 n 的分组码一共有 q^n 个可能的码字。从中选取包含 $M=2^k$ 个码字的子集构成一种码，这样可以把一个 k 比特的信息分组映射到所选择子集中的一个长度为 n 的码字。这样得到的分组码称为 (n,k) 码。

假设 C_i、C_j 是某种 (n,k) 码的两个任意的码字，α_1 和 α_2 是码元字符集中的两个任意的元素，那么当且仅当 $\alpha_1C_i+\alpha_2C_j$ 也是该分组码的码字时，我们称该分组码为线性的，叫做线性分组码。汉明（Hamming）码、哈达马（Hadamard）码和格雷（Golay）码是实际应用中常见的线性分组码。

循环码是线性分组码的一个重要的子集。循环码满足下列的循环移位特性：如果 $C=[c_{n-1}c_{n-2}\ldots c_1c_0]$ 是循环码的一个码字，那么所有 C 的循环移位都是该循环码的码字。由于循环码的循环特性，使得循环码具有很多构造上的特点，可以在编码和解码时加以利用。现在针对循环码已经设计出了许多有效的编码和硬判决译码算法，这样使含有大量码字的长循环分组码在实际系统中得到实用。循环汉明码、循环格雷码、最大长度移存器码和

BCH 码都是重要的循环码。

卷积码是将发送的信息序列通过一个线性的、有限状态的移位寄存器而产生的编码。通常卷积码的编码器由 K 级（每级 k 比特）移位寄存器和 n 个线性代数函数发生器组成，需要编码的二进制数据串行输入移位寄存器，每次移入 k 比特数据。每个 k 比特的输入序列对应一个 n 比特的输出序列。

另外为了尽可能的把传输过程中产生的错误均匀的分散到不同的码字中去，还会采用称为“交织”的手段。交织也主要分为两类：块交织和卷积交织。

在较为恶劣的信道情况下，如无线信道和电力线信道，单独使用一种信道编码不能达到满意的效果。人们又提出了级联码的概念，级联码也分为两种：串联级联码和并联级联码。串联的级联码由内码和外码组成，一般分别采用线性分组码和卷积码，译码时按次序先后译码。并联的级联码则没有内外码之分，在译码时一般采用反馈迭代算法，同时进行译码，也称为 Turbo 码。

4. COFDM 系统模型

为了提高 OFDM 系统的性能，进一步降低系统的误码率，人们在 OFDM 系统中加入信道编码，这样的 OFDM 系统一般称为 COFDM（Coded OFDM）系统。

在 COFDM 系统中，信道编码模块一般都包含两个主要部分：前向纠错码（FEC）和交织（Interleaving）。前向纠错码的作用是纠正信道产生的错误，恢复正确的原始信息；交织的作用是把信道中的突发错误打散，尽可能均匀地分散到接收到的数据序列里面，提高系统抗突发性错误的能力。

13.4.2　卷积码原理及设计

1. 卷积码编码

卷积码是将发送的信息序列通过一个线性的，有限状态的移位寄存器而产生的编码。通常卷积码的编码器由 K 级（每级 k 比特）的移位寄存器和 n 个线性代数函数发生器（这里是模 2 加法器）组成，如图 13-27 所示。

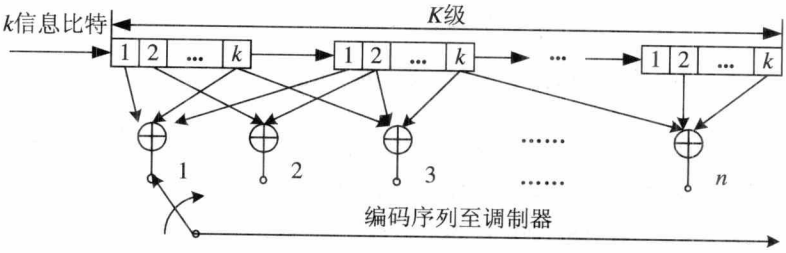


图 13-27　卷积码的编码器

需要编码的二进制数据串行输入移位寄存器，每次移入 k 比特数据。每个 k 比特的输入序列对应一个 n 比特的输出序列。因此卷积码的编码效率定义为 $R_c = k / n$。参数 K 被称作卷积码的约束长度，它表示当前的 n 比特输出序列与多少个 k 比特输入序列有关系，同

时也是一个决定编码复杂程度的重要参数。

描述卷积码的方法之一是给出其生成矩阵。一般说来卷积码的生成矩阵是一个半无限矩阵，因为输入序列是半无限长度的。

另一种描述卷积码的方法是用一组 n 个矢量——称为卷积码的生成多项式来表示。对应于 n 个代数函数发生器（这里是模 2 加法器）与移位寄存器的连接方式，我们得到 n 个生成多项式，这里是 n 个长度为 $K\times k$ 的矢量。某个矢量的第 i 个元素为 1，表示寄存器相应的位置与该矢量对应的模 2 加法器相连；反之，如果该元素为 0，则表示寄存器相应的位置与该矢量对应的模 2 加法器不连接。

另外三个关于卷积码的重要描述是树图、网格图和状态图。

树图以带有分支的树的形式表示编码器的结构：树根表示编码器的初始状态，通常都是全零状态“00……00”；树的分支根据输入的序列产生，分别代表编码器在相应输入下的后续状态和输出序列。

网格图是由树图变化而来的。仔细观察树图可以发现，当树的长度超过约束长度 K 时，树图的结构就出现了重复的现象——具有相同状态的两个节点所发出的所有分支具有相同的结构和输出。这说明超过约束长度之后，具有相同状态的节点可以合并。通过在树图中作节点合并我们就获得了比树图更加紧凑的网格图。在网格图上可以直观地表示和分析编码和译码的过程。

状态图比网格图更为紧凑，它表明了编码器可能存在的状态，以及各状态间可能存在的转移路线。在状态图上还标有状态转移的条件及相应的编码器输出。

下面用一个具体的例子进行讨论。如图 13-28 所示的为约束长度 $K=3$，$k=1$ 和 $n=2$（$R_c=1/2$）的二进制卷积码。

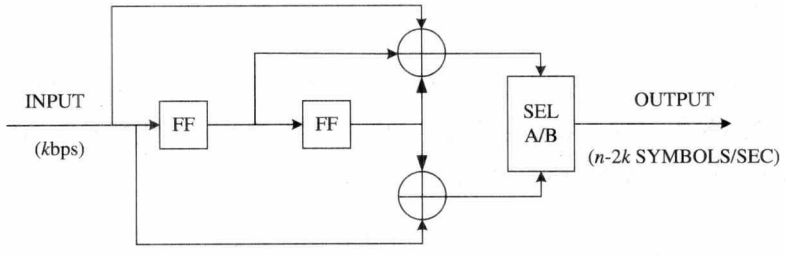


图 13-28　K=3，k=1 和 n=2（R_c=1/2）的卷积码编码器

假设移位寄存器的状态是全零，如果第一个输入比特是 1，那么 2 比特的输出序列是 11。如果第二个输入比特是 0，那么输出序列是 10，以此类推。将两个输出比特自上到下编号为 1 和 2，那么根据图中移位寄存器与模 2 加法器的连接情况，可以写出此卷积码的生成多项式：$g_1=[111]$，$g_2=[101]$。该成多项式还可以用 8 进制表示成(7,5)，这种方式更为简练。

下面给出图 13-28 卷积编码器产生的树图，如图 13-29 所示。

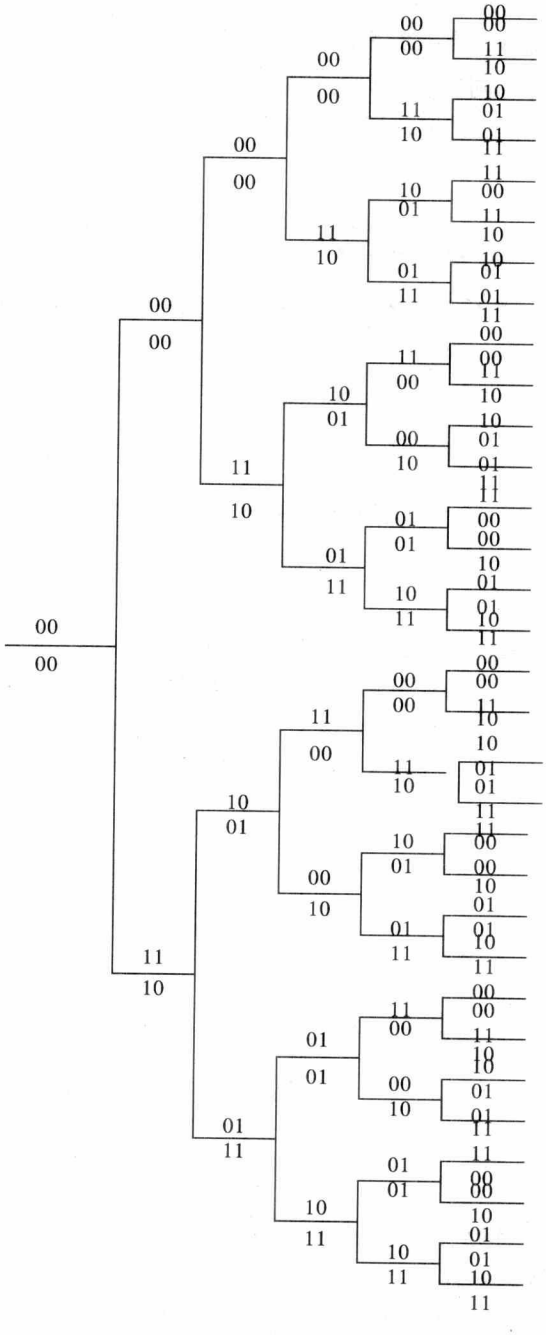


图 13-29　K=3，k=1 和 n=2（ R_c =1/2 ）的卷积码树图

在树图中，我们一般约定如果下一个输入比特为“0”，则取上分支；若下一个输入比特为“1”，则取下分支。

仔细观察图 13-29，我们不难发现第三级之后的结构是自身结构的重复。这个特征与约束长度 K =3 相对应。这说明每一级的 2 比特输出序列取决于当前输入的一个比特和先

前输入的两个比特，即移存器中的两个比特。我们把树图中的每个节点写上对应于移存器可能状态的标记，不难发现，具有相同标记（相同状态）的节点产生的分支具有相同的结构，即分支上的节点具有相同的状态分布和输出序列。在这个意义上，我们可以认为这样的两个节点是完全相同的。那么把这样的节点进行合并，我们就得到了更为紧凑的网格图，如图 13-30 所示。

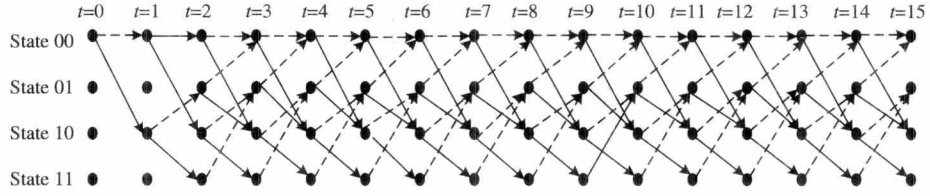


图 13-30　K=3，k=1 和 n=2（R_c =1/2）的卷积码网格图

在画网格图时，我们一般用实线和虚线来区分输入比特是“0”还是“1”。在本文中，我们约定用虚线表示输入比特为“0”时的状态转移，用实线表示输入比特为“1”时的状态转移。

仔细观察图 13-30，我们会看到经过一开始的过渡阶段后，网格图的每一级都有 4 个节点，分别对应移位寄存器的 4 个状态，每个节点都有两条进入的路径和两条出去的路径。而且在第二级之后，网格图的每一级都和下一级重复。我们将网格图的某一级提取出来，就是该卷积编码器的状态图，如图 13-31 所示。

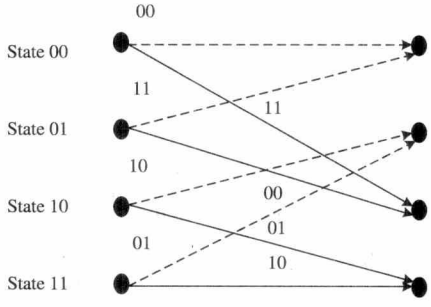


图 13-31　K=3，k=1 和 n=2（R_c =1/2）的卷积码状态图

在状态图中，我们依然约定用虚线表示输入比特为“0”时的状态转移，用实线表示输入比特为“1”时的状态转移。

在实际编码过程中，我们不难发现，最后的 K -1 个输入的 k 比特信息，无法影响到与其之前输入的 k 比特信息相同数量的 n 比特输出。为了让最后的 K -1 个输入比特也能影响到同样多的 n 比特输出，我们需要多输出 K -1 个 n 比特输出。这可以通过继续输入 K -1 个 n 比特的“00……00”来实现。这在编码中被称作“Flushing”。如果不采取这种处理方式，最后输出的那些编码结果会失去原有的纠错性能。在突发模式的工作环境中，这也提供了一个固定的编码结束状态，只要在每次触发前清空移存器，我们就可以把编码器的开始状态和结束状态都设为零。对于译码器正确地译码提供保证。

2. Viterbi 译码

Viterbi 译码是卷积码的最佳译码算法。Viterbi 译码算法是 Andrew J. Viterbi 提出的卷积码译码算法，他关于此算法的论文于 1967 年发表在 IEEE Transactions on Information Theory 上，题目是“Error Bounds for Convolution Codes and An Asymptotical Optimum Decoding Algorithm”。

Viterbi 译码算法的优点是固定的译码延时，并且适合硬件实现。但是它对计算量和存储量的要求随着约束长度 K 成指数增长，所以对 Viterbi 算法的应用一般局限于 $K \leqslant 9$ 的场合。

Vitetbi 译码算法可以描述如下：

把在时刻 i 状态 S_j 所对应的网格图节点记作 $S_{j,i}$，每个网格节点被分配一个最小不完全路径长度值 $V(S_{j,i})$。节点值按如下方式计算：

① 设 $V(S_{0,0})=0$，$i=1$。

② 在时刻 i，对于进入每一个节点的所有路径计算其不完全路径长度。

③ 令 $V(S_{j,i})$ 为 i 时刻到达与状态 S_j 相对应的节点 $S_{j,i}$ 的最小不完全路径长度。该路径为留存路径，其余非留存路径将从网格图中删除。以这种方式可以从 $S_{0,0}$ 处生成一组最小路径。

④ L 表示输入的待编码信息序列的符号长度，其中每个符号为 k 比特，m 为编码器中移存器长度，若 $i<L+m$，令 $i=i+1$，返回②。

⑤ 一旦计算出所有的节点值，则从 $i=L+m$ 时刻沿网格图中留存路径里具有最小路径长度的支路回溯即可得到译码序列。

关于不完全路径长度，硬判决时采用 Hamming 距离，软判决时采用 Euclidean 距离。

下面我们以 $K=3$，$k=1$ 和 $n=2$（$R_c=1/2$）的二进制卷积码为例说明 Viterbi 译码算法的具体过程。在这个例子中，为了简单起见，计算最小不完全路径采用 Hamming 距离，也就是硬判决。

如图 13-32 所示为该编码器的一次编码过程的示意图，我们选择的输入序列为 15 比特的信息——“010111001010001”，以及 2 比特的“Flushing”信息——“00”。图中的实线表示编码过程中的状态转移，即实际的编码路径。由图中可以看出，编码过程始于状态“00”，同时也结束于状态“00”。

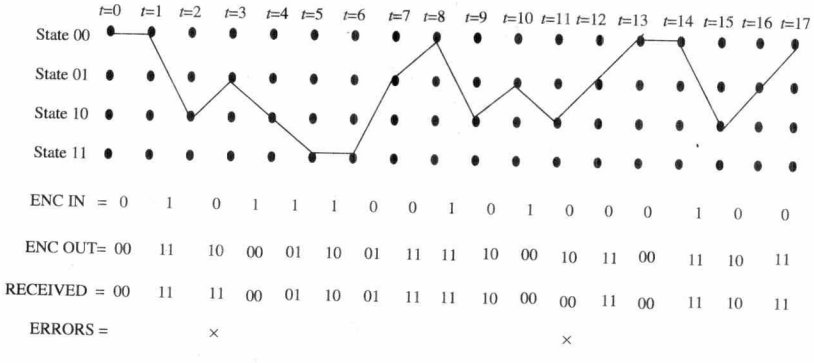


图 13-32 K=3，k=1 和 n=2（R_c=1/2）的二进制卷积码编码过程

图 13-32 中还在编码路径的下面标出了编码输入序列，编码输出序列和接收端接收到的含有错误的序列，并且标出了错误的位置。

在图 13-32 中的每个符号间隔，我们会接收到 2 比特的符号。我们用一个累计错误矩阵（Accumulated Error Metric）来存放计算得到的最小不完全路径值，即我们接收到的符号序列与所有可能接收到的符号序列之间的“距离”。我们使用硬判决的 Hamming 距离，就是计算接收到的符号序列与所有可能接收到的符号序列之间不同比特的个数。对于每一个分支，这个距离只可能是“0”、“1”或“2”。从第二个符号间隔开始，该矩阵的值为前一个符号间隔的矩阵值与当前分支的“距离”之和。

如图 13-33 所示，当 $t=1$ 时，我们收到了“00”，此时所有可能收到的符号是“00”和“11”。不难算出，它们之间的 Hamming 距离分别为 0 和 2。因为此时只有两个节点存在可能的路径，所以累计错误矩阵的另外两个元素没有定义。如果此时判决的话，就应该选择 $t=1$ 时状态为“00”的节点开始回溯，得到正确的译码结果“0”。

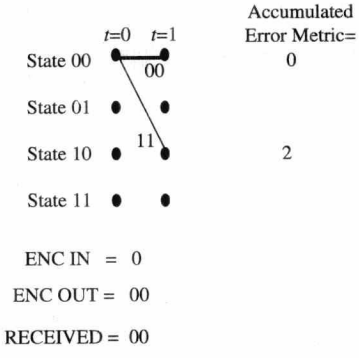


图 13-33 $t=1$ 时的情况

图 13-33 中的粗实线表示正确的回溯路径，即译码路径。所有可能的路径都在图中标出，下面的几幅图也是如此。

当 $t=2$ 时，情况变得复杂起来，如图 13-34 所示。我们得到了 4 条可能的路径，必须计算这四条路径与接收符号“11”之间的距离。

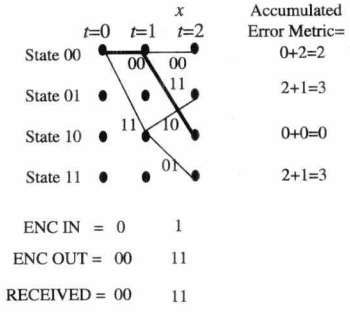


图 13-34 $t=2$ 时的情况

图 13-34 中标明了此时累积错误矩阵中每一个状态对应的最小不完全路径值。

图 13-35 是 $t=3$ 时的情况，比 $t=2$ 时的情况更为复杂。这时，我们有 8 条可能的路径，每个状态都是两条。在计算累计错误矩阵时要进行四次比较，选择不完全路径值最小的那一条路径。

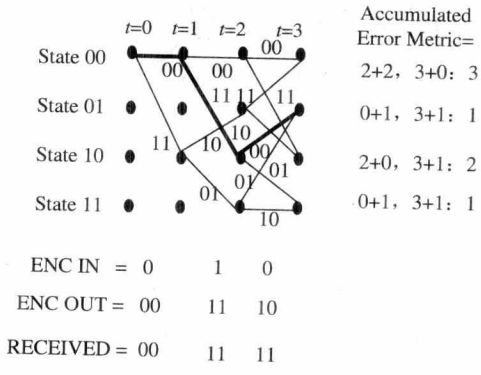


图 13-35　$t=3$ 时的情况

从图 13-35 中可以看出，如果此时进行判决就产生了麻烦：有两条路径具有相同的最小不完全路径值，都为“1”。这时只好任取一条路径进行译码，也就是说有 50%的错误概率。不过我们发现，无论选择哪一条路径，回溯到 $t=2$ 的时刻，都到达了同一个节点“10”。这说明即使沿着错误的路径回溯，也会回到正确的译码路径上来。这就保证了在这种情况下，错误会得到控制而不是扩散。在开始的错误后，我们会很快回到正确的译码路径。

下面我们再看 $t=4$ 时的情况，如图 13-36 所示。

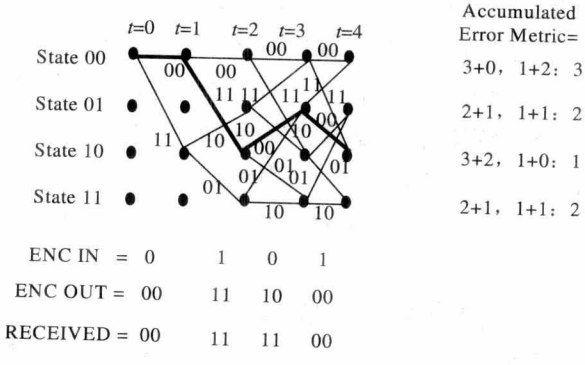


图 13-36　$t=4$ 时的情况

由图 13-36 中可以看出，$t=4$ 时计算的复杂度没有增加，和 $t=3$ 时相同。而且我们发现，如果此时判决，就能够修正 $t=3$ 时判决可能产生的错误。

$t=5$ 时的情况如图 13-37 所示。接下去的情况也是一样，当 $t=17$ 时，如果只保留粗实线，我们就会得到和图 13-32 完全一样的路径。

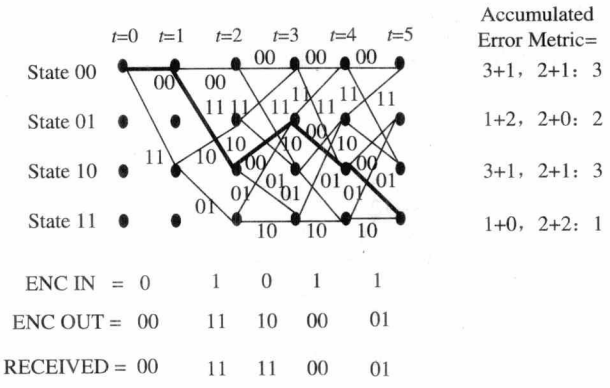


图 13-37　$t=5$ 时的情况

一般来说，参数为 (K,k,n) 的卷积码将产生 $2^{k(K-1)}$ 个状态的网格图。用 Vitetbi 译码算法对其进行译码时，要保存 $2^{k(K-1)}$ 条留存路径和 $2^{k(K-1)}$ 个最小路径度量。在网格图每一级的每一个节点都要计算 2^k 条可能路径的度量。这样在执行每一级的译码时，计算量将随着 K 与 k 呈指数增加。所以 Vitetbi 译码算法不光要局限于 K 值较小的场合，同时要求 k 的值也比较小。

13.4.3　交织原理及设计

在具有多径和衰落特点的信道里，由时变的多径传输造成的信号衰落常常导致信号电平小于噪声电平，结果是出现连续的大量错误，这种错误具有突发特征。人们已经研究并构造出了能够纠正突发错误的编码，如 Fire 码和 Burton 码等。

处理信道产生的突发错误还有一种有效的方法，就是对编码后的数据进行交织，把突发错误信道人为地转变为统计独立错误信道。编码后的数据经交织器重新排序后在信道上传输。在接收端解调后，解交织器将数据恢复到原来的顺序后送入译码器。由于交织/解交织的效果，信道中产生的突发性错误在时间上得到分散。

交织的方式有两种：块交织（Block Interleaving）和卷积交织（Convolutional Interleaving）。

块交织的过程是：阶交织器将编码后的数据按行的方向排成 m 行 n 列的阵列形式，然后按列的方向依次读出数据，完成数据的交织。在接收端，解交织器将解调后的数据按列的方向排成 m 行 n 列的阵列形式，然后按行的方向依次读出数据，完成数据的解交织。

卷积交织器的结构和原理在 Ramsey 和 Forney 的著作中有所论述，本文此处不做介绍。

13.5　OFDM 通信系统设计

13.5.1　发射机设计

本系统设计的发射机的框图如图 13-38 所示。

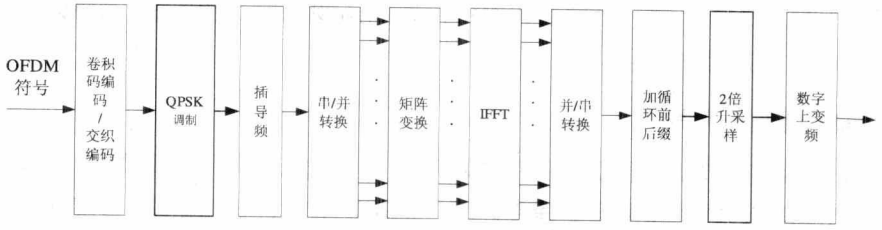


图 13-38　OFDM 通信系统发送机框图

1. 信道编码

信道编码采用卷积编码和交织编码进行信道级联编码。卷积编码码率为 1/2，仿真时设置 $k=1$，$G=[1\ 0\ 1\ 1\ 0\ 1\ 1;1\ 1\ 1\ 1\ 0\ 0\ 1]$，将输入的 90 个 0、1 二进制数经过卷积编码后可得到 192 个 0、1 二进制数。交织编码采用 24 行 8 列的矩阵，按行写入，按列读出，交织编码可以有效地抗突发干扰。

2. QPSK 调制

在数字信号的调制方式中，使用了 QPSK（四相移键控），这种调制方式具有较高的频谱利用率以及较强的抗干扰性，在电路上实现也较为简单，而且具有较好的 PAPR 抑制性能。QPSK 调制的映射方式如表 13-1 所示。

表 13-1　QPSK 调制数据

输入数据	调制后数据
1,1	1+*i*
1, -1	1-*i*
-1, 1	-1+*i*
-1, -1	-1-*i*

3. 插导频

导频数据是在进行矩阵变换之前插入有效数据的，在系统设计中我们每 8 个有效数据插入一个导频，但是数据中间位置不插入导频。96 个复数据插入 10 个导频之后，一帧数据长度为 106。

4. 矩阵变换

矩阵变换模块是为了降低系统的 PAPR，采用方法的是前面介绍的改进 Nyquist 脉冲整形法（PS）。这里的矩阵大小为 106×128，滚降系数 $\alpha=0.22$。通过这种方法，可以显著地改善 OFDM 通信系统的 PAPR 分布，大大降低了峰值信号出现的概率以及对功率放大器的要求，节约成本。在接收端恢复原始信号只需要在 FFT 运算之后乘上一个发端矩阵的逆矩阵即可。

5. IFFT 变换

经过矩阵乘模块后，一帧数据长度为 128，由于子载波个数为 256，所以需要在数据

后面补 128 个零。补零之后，考虑到频谱利用率的问题需要对数据进行搬移（索引为 1～64 的数据搬移到数据最后）。

6. 加循环前后缀与升采样

用 IFFT 输出的数据的前 32 点作为循环后缀，后 32 点作为循环前缀。假定射频的采样时钟为 2.56MHz，所以需要数据速率匹配，对基带信号进行升采样。升采样过程由两部分组成。第一部分对加了循环前后缀之后的数据进行 2 倍的升采样，所采取的方式是在每个数据中间插入 1 个 0；第二部分用上变频模块的 CIC 内插滤波器对信号进行 20 倍升采样。

7. 数字上变频

数字上变频完成的功能是将基带信号进行线性频谱搬移，实质上就是将基带成形信号（I、Q 两个支路）乘以一个载波信号（同样分为 I、Q 两个支路），再把两个支路相加即可。但为了抑制已调信号的带外辐射，在同相和正交支路上还分别增加一个具有线性相位特性的低通成型滤波器 FIR。另外，为了使产生的基带信号与后面的采样速率相匹配，在进行正交调制前还必须通过 CIC 内插滤波器将基带信号进行 20 倍升采样处理，整个实现过程如图 13-39 所示，数字上变频模块中包含了基带成型滤波器、梳状内插滤波器和数控振荡器。

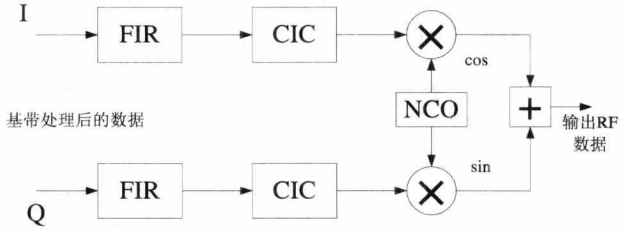


图 13-39　数字上变频实现结构

- FIR 滤波器

由于在基带信号送往数字上变频器之前，经过 2 倍升采样，所以频谱产生了两次的镜像，需要用一个基带滤波器滤除带外的杂散频率。此数字上变频模块中的基带成型滤波器采用 FIR 低通滤波器来实现。FIR 具体实现结构如图 13-40 所示。

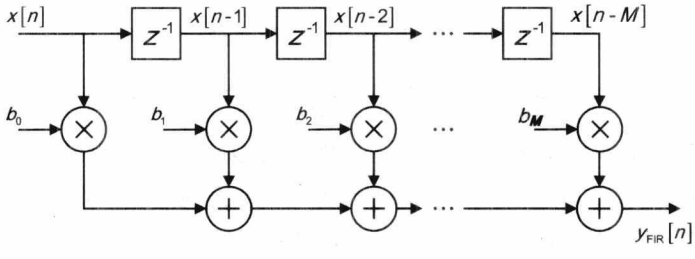


图 13-40　FIR 滤波器实现结构

综合考虑系统的需要和资源的占用，为了达到性能指标（抽样截止频率为 128kHz，通带截止频率为 20kHz，阻带截止频率为 40kHz，带内纹波动小于 1dB，带外衰减 100dB），

经 MATLAB/Simulink 工具箱设计出 FIR 滤波器的阶数为 19 阶。各抽头系数如表 13-2 所示，FIR 滤波器的幅频特性如图 13-41 所示。

表 13-2　3 阶 FIR 滤波器的抽头系数表

抽头	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
系数	0.0017	0.0102	0.0255	0.0288	-0.0059	-0.0601	-0.0496	0.0914	0.2964	0.3956
抽头	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	
系数	0.2964	0.0914	-0.0496	-0.0601	-0.0059	0.0288	0.0255	0.0102	0.0017	

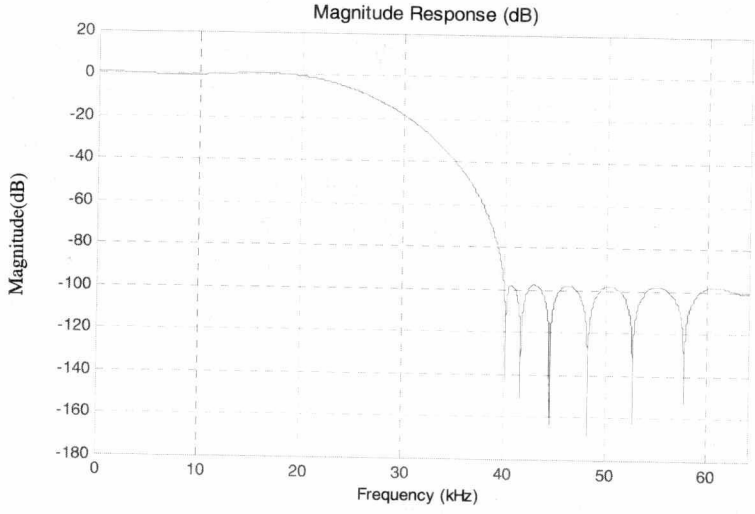


图 13-41　19 阶 FIR 滤波器的幅频特性曲线

• CIC 内插滤波器

由于射频的采样频率需要与射频端进行速率匹配，在上变频之前还需要对数据进 20 倍升采样。在这个阶段升采样使用的是 CIC 内插滤波器，它是由 E.B.Hogenauer 首先提出来的一种级联积分梳状滤波器，也称为 Hogenauer 滤波器，主要用于高采样率转换的滤波器设计中。

整个 CIC 内插滤波器的传递函数是所有梳状滤波器和积分滤波器共同作用的结果，其实现结构图如图 13-42 所示。N 级 CIC 内插滤波器的传递函数为：

$$H(z)=\boldsymbol{H}_I^N(z)\boldsymbol{H}_C^N(z)=\frac{\left(1-z^{-RM}\right)^N}{\left(1-z^{-1}\right)^N}=\left(\sum_{k=0}^{RM-1}z^{-k}\right)^N \qquad (式\ 13\text{-}69)$$

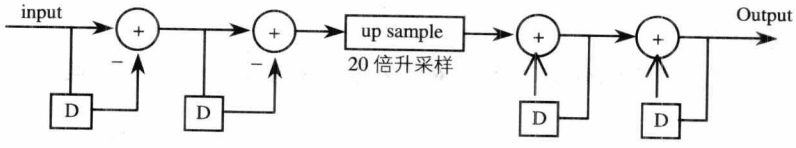


图 13-42　CIC 内插滤波器实现结构

本文设计的CIC 内插滤波器的参数取为 $R=20, M=1, N=2$，其幅频特性如图 13-43 所示。

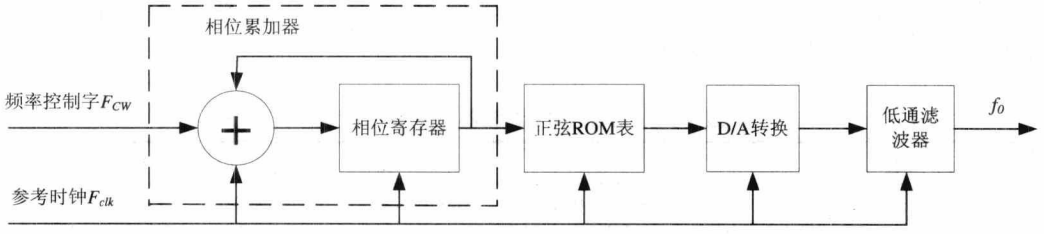


图 13-43　DDS 原理框图

• 直接数字频率合成器

数控振荡器采用的是直接数字频率合成器（Direct Digital Sythesis，DDS）的方式来完成的。DDS 具有超高速的频率转换时间、极高的频率分辨率和较低的相位噪声，在频率改变与调频时，DDS 能够保持相位的连续，因此很容易实现频率、相位和幅度调制。

DDS 的原理框图如图 13-43 所示。图中相位累加器可在每一个时钟周期来临时将频率控制字所决定的相位增量 M 累加一次，如果计数大于累加器位宽则自动溢出，而只保留后面的 N 位数字于累加器中。正弦查询表 ROM 用于实现从相位累加器输出的相位值到正弦幅度值的转换，然后送到 D/A 中将正弦幅度值的数字量转变为模拟量，最后通过滤波器输出一个很纯净的载波信号。

13.5.2　接收机设计

本系统设计的接收机的框图如图 13-44 所示。

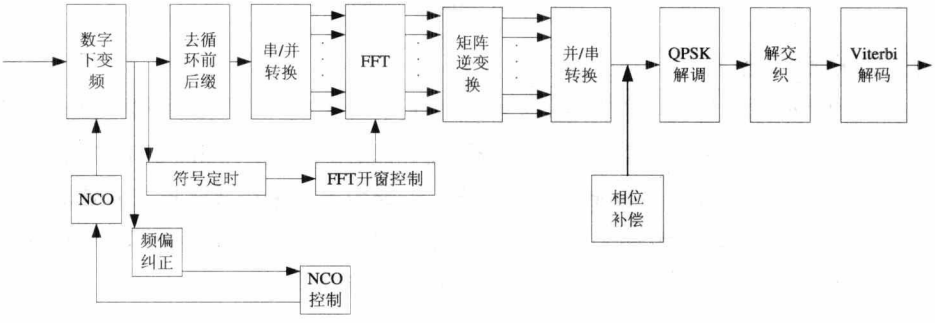


图 13-44　OFDM 系统接收机框图

接收机很多通信处理的模块都是与发射机的相关模块功能相似，这里不再一一介绍，接收机主要增加了同步模块。

• 系统同步

本文在前面已经详细介绍了本系统所采用的同步算法和同步处理过程，下面再简单做个概括。

将基带信号送到帧检测模块，这一步的目的是系统需要知道是否有信号到达和信号来到后粗定时的位置，当检测到信号来到后同时启动粗定时模块、频偏捕获模块。符号粗定时与粗频偏捕获利用的是前导序列中的短序列来实现的，而细频偏估计采用的是前导序列

中的长训练序列实现的。然后，利用前导序列中的 T_{1m} 和 T_{2m} 进行本地互相关，得到相关峰值，通过峰值所确定的位置确定精确的 FFT 开窗位置，并通过 OFDM 数据帧中的循环前后缀的循环特性进行频率跟踪，在频域中，再利用解调出的导频信息进行相位补偿。

13.5.3　系统仿真参数

根据前面已经介绍的 OFDM 通信系统的设计，表 1-3 给出了仿真系统的主要参数。

表 13-3　OFDM 通信系统仿真参数

数据传输速率	19.2Kbps
调制方式	QPSK
纠错编码	卷积编码/交织编码
译码	Viterbi 译码
OFDM 符号长度	320
保护间隔	循环前后缀各 32 点
子载波数	128
IFFT/FFT 点数	256
载波频率	320 kHz
信道	加性高斯白噪声（AWGN）

13.5.4　系统性能仿真

根据前面介绍的 OFDM 通信系统的设计，可以仿真得到设计系统的性能如图 13-45 所示。

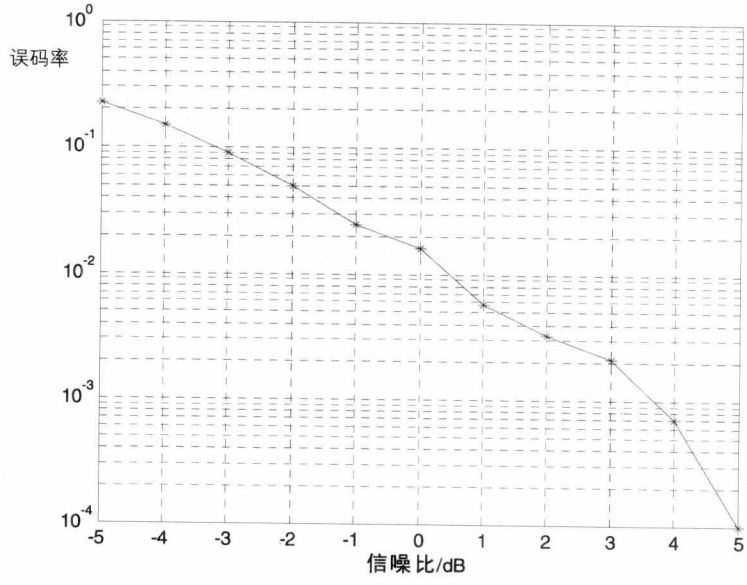


图 13-45　OFDM 通信系统性能仿真

13.6 OFDM 通信系统仿真程序

下面列出 OFDM 通信系统 MATLAB 的部分仿真程序以及注释，供读者参考分析。

```
%main_OFDM.m
%本 OFDM 通信系统的仿真设计，包括编码、调制、IFFT、
%上下变频、高斯信道建模、FFT、PAPR 抑制、各种同步、解调和解码等模
%块，并通过系统性能的仿真验证了系统设计的可靠性。

clear all
close all
clc

%+++++++++++++++++++++++++++全局变量+++++++++++++++++++++++++++++++
% seq_num                  表示当前帧是第几帧
% count_dds_up             上变频处的控制字的累加
% count_dds_down           下变频处的控制字的累加（整数）
% count_dds_down_tmp       下变频处的控制字的累加（小数）
% dingshi                  定时同步的定位
% m_syn                    记录定时同步中的自相关平台
global seq_num
global count_dds_up
global count_dds_down
global count_dds_down_tmp
global dingshi
global m_syn
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% SNR_Pre                  设定用于仿真的信噪比的初值
% interval_SNR             设定用于仿真的信噪比间隔
% frame_num                每一个信噪比下仿真的数据帧数
% err_int_final            用于计算每帧出现的误比特数
% fwc_down                 设定的接收机初始载波频率控制字
% fre_offset               设定接收机初始载波频率偏移调整量（单位为 Hz）
% k0                       每次进入卷积编码器的信息比特数
% G                        卷积编码的生成矩阵
SNR_Pre=-5;
interval_SNR=1;

for SNR_System=SNR_Pre:interval_SNR:5

frame_num=152;
dingshi=250;
err_int_final=0;
fwc_down=16.050;
fre_offset=0;
k0=1;
```

```
G=[1 0 1 1 0 1 1;1 1 1 1 0 0 1 ];

disp('--------------start------------------');

for seq_num=1:frame_num, %frame_num 帧数

%++++++++++++++++++++++以下为输入数据部分++++++++++++++++++++++
datain=randint(1,90);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++以下为信道卷积编码部分++++++++++++++++++++
encodeDATA=cnv_encd(G,k0,datain);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++信道交织编码++++++++++++++++++++++++
interlacedata=interlacecode(encodeDATA,8,24);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++以下为 QPSK 调制部分++++++++++++++++++++
QPSKdata=qpsk(interlacedata);
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++生成训练序列++++++++++++++++++++++++
if seq_num<3
trainsp_temp=seq_train();
end
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++插入导频++++++++++++++++++++++++++
PILOT=(1+j);
m_QPSKdata=QPSKdata;
data2fft_temp=[m_QPSKdata(1:8),PILOT,m_QPSKdata(9:16),PILOT,m_QPSKdata(
17:24),PILOT,m_QPSKdata(25:32),PILOT,m_QPSKdata(33:40),PILOT,m_QPSKdata(41:
48),m_QPSKdata(49:56),PILOT,m_QPSKdata(57:64),PILOT,m_QPSKdata(65:72),PILOT
,m_QPSKdata(73:80),PILOT,m_QPSKdata(81:88),PILOT,m_QPSKdata(89:end)];
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

trainsp_temp2=[trainsp_temp,zeros(1,128)];
trainsp=[trainsp_temp2(65:256),trainsp_temp2(1:64)];

%++++++++++++++++++++++++降 PAPR 矩阵变换++++++++++++++++++++++
matix_data=nyquistimp_PS();
matrix_mult=data2fft_temp*matix_data;
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

data2fft2=[matrix_mult(65:128),zeros(1,128),matrix_mult(1:64)];

%++++++++++++++++++++++++++ifft 运算++++++++++++++++++++++++++
if seq_num==1
   ifftin=trainsp;
```

```
elseif seq_num==2
    ifftin=trainsp;
else
    ifftin=data2fft2;
end
IFFTdata=fft_my(conj(ifftin)/256);
IFFTdata=conj(IFFTdata);
% figure
% plot(real(IFFTdata))
% xlabel('realIFFTdata')
% figure
% plot(imag(IFFTdata))
% xlabel('imagIFFTdata')
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++以下为插入循环前后缀，2 倍升采样+++++++++++++++
data2fir=add_CYC_upsample(IFFTdata,2);
% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% ++++++++++++++++++++++++++fir 低通滤波++++++++++++++++++++++++++
guiyi_a=[0.0017216  0.010162    0.025512    0.028801    -0.0059219
-0.060115   -0.0496 0.091431    0.29636 0.3956  0.29636 0.091431
-0.0496 -0.060115   -0.0059219  0.028801    0.025512    0.010162
0.0017216 ];
%抽样截止频率为 128kHz,通带截止频率为 20kHz,阻带截止频率为 40kHz,带内纹波动小于 1dB,
带外衰减 100dB
txFIRdatai=filter(guiyi_a,1,real(data2fir));
txFIRdataq=filter(guiyi_a,1,imag(data2fir));
% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++发射机 cic 滤波++++++++++++++++++++++++++
CICidatai=cic_inter(txFIRdatai,20);
CICidataq=cic_inter(txFIRdataq,20);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++++++上变频++++++++++++++++++++++++++++++
fwc_up=16;                      %控制字可以选择
DUCdata=up_convert_ofdm(fwc_up,CICidatai,CICidataq);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%++++++++++++++++++++++++++高斯白噪声信道++++++++++++++++++++++++
[DUCdata,datamax]=guiyi_DUCdata(DUCdata);
awgn_data=awgn(DUCdata,SNR_System);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%*****************************接受机*****************************
%++++++++++++++++++++++++++++++下变频++++++++++++++++++++++++++++++
DUCdata_tmp=awgn_data;
fwc_down=fwc_down+(fre_offset*128/2560000);
r_fre_offset=2560000*((fwc_down-fwc_up)/128);
```

```
[DDCdatai,DDCdataq]=down_convert_ofdm(fwc_down,DUCdata_tmp);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++接收机 cic 滤波+++++++++++++++++++++++++++
CICddatai=cic_deci(DDCdatai,40,40);
CICddataq=cic_deci(DDCdataq,40,40);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++++fir 低通滤波+++++++++++++++++++++++++
guiyi_b=[ 0.019527  -0.03934    0.049055    -0.018102   -0.1003 0.5944
0.5944  -0.1003 -0.018102   0.049055    -0.03934    0.019527];
%抽样截止频率为 64kHz，通带截止频率为 20kHz，阻带截止频率为 30kHz，带内纹波动小于 1dB，
带外衰减 60dB
rxFIRdatai=filter(guiyi_b,1,CICddatai);
rxFIRdataq=filter(guiyi_b,1,CICddataq);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++++量化+++++++++++++++++++++++++++
q_rxFIRdatai=sign(rxFIRdatai);
q_rxFIRdataq=sign(rxFIRdataq);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++定时同步检测+++++++++++++++++++++++++
if seq_num<3
time_syn(q_rxFIRdatai,q_rxFIRdataq);
end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++++频率同步+++++++++++++++++++++++++++
fre_offset=fre_syn(rxFIRdatai,rxFIRdataq);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++++fft 运算+++++++++++++++++++++++++++
if seq_num>2
   seq_num-2
   fftw=32+dingshi;
   rxFIRdata_syn=rxFIRdatai(fftw:fftw+255)+j*rxFIRdataq(fftw:fftw+255);
   FFTdata=fft_my(rxFIRdata_syn);
   %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++降 PAPR 逆矩阵变换+++++++++++++++++++++++
fftdata_reg=[FFTdata(193:256),FFTdata(1:64)];
dematrix_data=fftdata_reg*pinv(matix_data);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++++相位补偿+++++++++++++++++++++++++++
rx_qpsk_din_th=phase_comp(dematrix_data);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%+++++++++++++++++++++++++QPSK 解调部分+++++++++++++++++++++++++
```

```
    % figure
    % plot(rx_qpsk_din_th,'.')
    % xlabel('星座图')
    datatemp4=deqpsk(rx_qpsk_din_th);
    datatemp4=sign(datatemp4);
    for m=1:192
        if datatemp4(m)==-1
            datatemp4(m)=1;
        elseif datatemp4(m)==1
            datatemp4(m)=0;
        end
    end
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %++++++++++++++++++++++++++信道解交织++++++++++++++++++++++++++++
    interdout=interlacedecode(datatemp4,8,24);
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %++++++++++++++++++++以下为 viterbi 译码部分++++++++++++++++++++++
    decodeDATA=viterbi(G,k0,interdout);
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %++++++++++++++++++++++++++误比特统计++++++++++++++++++++++++++++
    err_final=sum(abs(decodeDATA-datain))
    err_int_final=err_int_final+err_final
    end
    end
    disp('-----------------------------------------------------');
    SNR_System
    err_rate_final((SNR_System-SNR_Pre)./interval_SNR+1)=err_int_final/(90*
(frame_num-2))
    disp('-----------------------------------------------------');

    end

    disp('--------------end-------------------');

    SNR_System=SNR_Pre:interval_SNR:5;
    figure
    semilogy(SNR_System,err_rate_final,'b-*');
    xlabel('信噪比/dB')
    ylabel('误码率')
    axis([-5,5,0,1])
    grid on
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


    %************************beginning of file*****************************
    %cnv_encd.m
    %卷积码编码程序
```

```
function output=cnv_encd(G,k0,input)
% cnv_encd(G,k0,input),k0 是每一时钟周期输入编码器的 bit 数,
% G 是决定输入序列的生成矩阵，它有 n0 行 L*k0 列，n0 是输出 bit 数，参数 n0 和 L
% 由生成矩阵 G 导出，L 是约束长度。L 之所以叫约束长度是因为编码器在每一时刻里输出序列
% 不但与当前输入序列有关，而且还与编码器的状态有关，这个状态是由编码器的前(L-1)k0。
% 个输入决定的，通常卷积码表示为(n0,k0,m)，m=(L-1)*k0 是编码器中的编码存储个数，
% 也就是分为 L-1 段，每段 k0 个有些人将 m=L*k0 定义为约束长度，有的人定义为
% m=(L-1)*k0 查看是否需要补 0，输入 input 必须是 k0 的整数部

%+++++++++++++++++++++++variables++++++++++++++++++++++++++
% G         决定输入序列的生成矩阵
% k0        每一时钟周期输入编码器的 bit 数
% input     输入数据
% output    输入数据
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

if rem(length(input),k0)>0
input=[input,zeros(size(1:k0-rem(length(input),k0)))];
end
n=length(input)/k0;
% 检查生成矩阵 G 的维数是否和 k0 一致
if rem(size(G,2),k0)>0
error('Error,G is not of the right size.')
end
% 得到约束长度 L 和输出比特数 n0
L=size(G,2)/k0;
n0=size(G,1);
% 在信息前后加 0，使存储器归 0，加 0 个数为(L-1)*k0 个
u=[zeros(size(1:(L-1)*k0)),input,zeros(size(1:(L-1)*k0))];
% 得到 uu 矩阵，它的各列是编码器各个存储器在各时钟周期的内容
u1=u(L*k0:-1:1);
%将加 0 后的输入序列按每组 L*k0 个分组，分组是按 k0 比特增加
%从 1 到 L*k0 比特为第一组，从 1+k0 到 L*k0+k0 为第二组，……,
%并将分组按倒序排列。
for i=1:n+L-2
u1=[u1,u((i+L)*k0:-1:i*k0+1)];
end
uu=reshape(u1,L*k0,n+L-1);
% 得到输出，输出由生成矩阵 G*uu 得到
output=reshape(rem(G*uu,2),1,n0*(L+n-1));
% ************************end of file**********************************


%************************beginning of file*****************************
%interlacecode.m

function dout=interlacecode(din,m,n)
%实现信道的交织编码
%din 为输入交织编码器的数据，m 与 n 分别为交织器的行列值
```

```
%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% din        输入数据
% m          交织器的行值
% n          交织器的列值
% dout       输出数据
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

for j=1:m
    temp(j,:)=din(j*n-(n-1):j*n);
end

dout_temp=reshape(temp,1,length(din));
dout=dout_temp(1:end);
%************************end of file*********************************


%************************beginning of file*****************************
%qpsk.m
%QPSK 调制映射
function dout=qpsk(din)

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% din        输入数据
% dout       输出数据
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

din2=1-2*din;
din_temp=reshape(din2,2,length(din)/2);
for i=1:length(din)/2,
   dout(i)=din_temp(1,i)+j*din_temp(2,i);
end
% ************************end of file*********************************


%************************beginning of file*****************************
%seq_train.m
%生成用于同步的训练符号
function dout=seq_train()
%第一帧产生短训练序列，第二帧产生长训练序列每个短训练符号由 16 个子载波组成，短训练序列
%是由伪随机序列经过数字调制后插 0 后，再经过 IFFT 之后得到的。具体过程如下：首先采用抽头
%系数为[1 0 0 1 ]的 4 级移位寄存器产生长度为 15 的伪随机序列之后末尾补 0，经过 QPSK 调
%制之后的伪随机序列只在 16 的整数倍位置上出现，其余的位置补 0，产生长度为 128 的序列，
%此序列再补 128 个 0 经过数据搬移后做 256 点的 IFFT 变换就得到 16 个以 16 为循环的训练序
%列，经过加循环前后缀就会产生 20 个相同的短训练序列。长训练序列的产生同短训练序列。

global seq_num

if seq_num==1
```

```
    fbconnection=[1 0 0 1];
    QPSKdata_pn=[m_sequence(fbconnection),0];
    QPSKdata_pn=qpsk(QPSKdata_pn);
elseif seq_num==2
    fbconnection=[1 0 0 0 0 0 1];
    QPSKdata_pn=[m_sequence(fbconnection),0];
    QPSKdata_pn=qpsk(QPSKdata_pn);
end

countmod=0;
for k=1:128
    if seq_num==1
        if mod(k-1,16)==0                    %生成 16 位循环的短训练符号
            countmod=countmod+1;
            trainsp_temp(k)=QPSKdata_pn(countmod);
        else
            trainsp_temp(k)=0;
        end
    elseif seq_num==2
        if mod(k-1,2)==0
            countmod=countmod+1;
            trainsp_temp(k)=QPSKdata_pn(countmod);
        else
            trainsp_temp(k)=0;
        end
    end
end

dout=trainsp_temp;
% ************************end of file**********************************


%************************beginning of file*****************************
%m_sequence.m
%用线性移位寄存器产生 m 序列

function [mseq]= m_sequence(fbconnection);

%++++++++++++++++++++++++variables+++++++++++++++++++++++++++++
% fbconnection   线性移位寄存器的系数
% mseq           生成的 m 序列
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

n = length(fbconnection);
N = 2^n-1;
register = [zeros(1,n - 1) 1];%定义移位寄存器的初始状态
mseq(1)= register(n);
for i = 2:N
    newregister(1)= mod(sum(fbconnection.*register),2);
    for j = 2:n,
```

```
        newregister(j)= register(j-1);
    end;
    register = newregister;
    mseq(i) = register(n);
end
% ************************end of file*********************************


%************************beginning of file*****************************
%nyquistimp_PS.m
%使用改进的 Nyquist 脉冲实现 OFDM 信号的 PAPR 抑制 function dout=nyquistimp_PS()
%改进的 Nyquist 脉冲整形方法能够显著改善 OFDM 信号的 PAPR 分布；该方法实现简单，
%和 PTS 和 SLM 相比无须迭代计算多个 IFFT 操作，无须传送边带信息，不会引起信号的畸变；
%通用性强，可以调整滚降系数以适应任何子载波数的通信系统。当然，Nyquist 脉冲成形的方法
%由于扩展了频谱，一定程度上降低了频谱利用率。

%creat a matrix to shape the subcarries.

N=106;
L=11;
b=0.22;

T=0.004;
Ts=T/N;
Bw=1/Ts;

begin=-Bw*(1+b)+Bw*(1+b)/128;
finish=Bw*(1+b)-Bw*(1+b)/128;
distance=Bw*(1+b)/64;
kk=0;
aa=log(2)/(b.*Bw);


for f=begin:distance:finish
    kk=kk+1;
    if abs(f)<=Bw*(1-b)
        spec=1;
       else if (abs(f)>Bw*(1-b))&(abs(f)<=Bw)
            spec=exp(aa.*(Bw.*(1-b)-abs(f)));
          else if (abs(f)>Bw)&(abs(f)<Bw*(1+b))
              spec=1-exp(aa.*(abs(f)-Bw.*(1+b)));
             else if  abs(f)>=Bw*(1+b)
                spec=0;
             end
          end
       end
    end
   C(kk)=spec;
end
```

```
for m=0:N-1
    for k=0:(N+2*L-1)
        p(m+1,k+1)=C(k+1)*exp(-i*2*pi.*m.*(k-L)./N);
    end
end

dout=p;
% ************************end of file*********************************
```

其他代码限于篇幅省略，具体请读者参见光盘内容。

13.7 本章小结

本章首先阐述了 OFDM 系统的基本原理，然后介绍了 OFDM 系统的 PAPR 抑制算法、同步算法以及编码算法设计，最后实现了 OFDM 通信系统的建模仿真设计。限于篇幅，许多模块的仿真程序没有在文中列出，请读者对照光盘内容进行完整的学习。

第 14 章　MIMO 通信系统仿真设计

OFDM 技术通过将频率选择性多径衰落信道在频域内转换为平坦信道，减小了多径衰落的影响。但用 OFDM 技术提高传输速率，就要增加带宽、发送功率和子载波数目，这对带宽和功率受限的无线通信系统是不现实的，子载波数目的增加也会使系统更为复杂。

MIMO 技术能够在空间中产生独立的并行信道来同时传输多路数据流，这样就有效地提高了系统的传输速率，即在不增加系统带宽的情况下增加频谱效率，但对于频率选择性深衰落依然是无能为力。

因此，将 OFDM 和 MIMO 两种技术相结合，就能达到两种效果。一种是实现很高的传输速率，另一种是通过分集实现很强的可靠性，从而很好地解决了两种技术单独使用时所面临的问题。

MIMO 和 OFDM 作为未来 4G 的核心技术，它们的结合在提高无线链路的传输速率和可靠性方面具有巨大潜力。本章对 MIMO 通信系统的原理与仿真技术进行系统的介绍。

14.1 MIMO 系统理论

多天线的概念最早是 Marconi 于 1908 年提出的。他最初的设想是利用多天线来抑制信道衰落，而现在的 MIMO 技术是通过增加发射天线和接收天线的数目，在收发两端之间形成多个并行传输信道，利用这些信道进行数据的并行传输来提高通信速率。同传统的单输入单输出（Single-input-single-output，SISO）信道相比，MIMO 信道可以提供更大的信道容量和更高的传输速率。MIMO 系统可以定义为一个在收发两端同时采用多天线技术的无线传输系统。真正意义上的 MIMO 技术研究是从 20 世纪 80 年代开始的，随后有更多的文章对它进行了理论上的分析，MIMO 技术开始成为无线通信技术研究中的热点。在未来的宽带无线通信系统中，MIMO 技术被认为是核心物理层技术之一。

MIMO 系统的核心思想就是：时间上的空时（Space-Time）信号处理同空间上的分集结合起来。时间上的空时信号处理一般是通过在发送端采用空时码（Space-Time Codes）来实现的，目前常见的空时码包括：空时分组码（Space-Time Blocks Codes）、空时格码（Space-Time Trellis Codes）、分层空时码（Layered Space-Time Codes）等。空时码的主要思想是利用空间和时间上的两维编码来实现一定的空间分集和时间分集，从而降低信道误码率，在空间上的分集是通过增加空间上的天线分布来实现的。这样做的好处是能把原来对用户来说是有害的无线电波的多径传播转变为对用户有利。最重要的一点是 MIMO 技术在多个方面提升无线通信系统性能的同时并不需要占用更多的无线带宽，所需要的仅仅是通信硬件设备和系统复杂度的增加。这项优点使得 MIMO 技术在频谱资源紧缺的现状下变

得更加吸引人。

14.1.1　MIMO 系统模型

假定一个点对点的 MIMO 系统有 n_T 根发射天线、n_R 根接收天线，采用离散时间的复基带线性系统模型描述，系统框图如图 14-1 所示。用 $n_T \times 1$ 的列向量 $\boldsymbol{x}$ 表示每个符号周期内的发射信号，其中第 i 个元素 x_i 表示第 i 根天线上的发射信号。

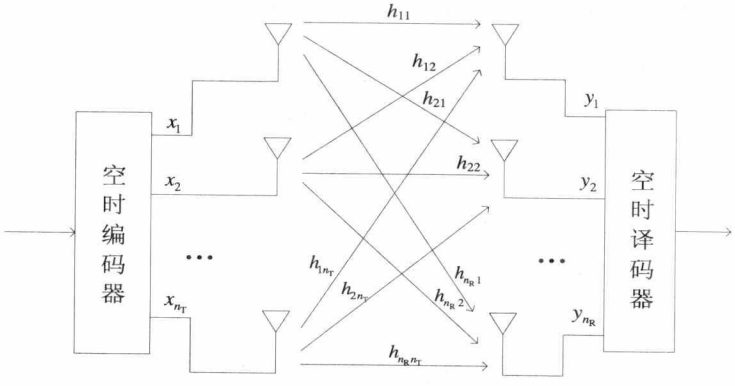


图 14-1　MIMO 系统框图

对于高斯信道，按照信息论，发射信号的最佳分布也是高斯分布。因此，$\boldsymbol{x}$ 的元素是零均值独立同分布的高斯变量。发射信号的协方差矩阵为：

$$\boldsymbol{R}_{xx} = E\left\{\boldsymbol{x}\boldsymbol{x}^H\right\} \tag{式 14-1}$$

其中，$E\{\cdot\}$ 代表均值；$\boldsymbol{A}^H$ 表示矩阵 $\boldsymbol{A}$ 的厄米特（Hermitian）转置矩阵，即 $\mathbf{A}$ 的复共轭转置矩阵。不管发射天线数 n_T 为多少，总的发射功率限制为 P，可以表示为：

$$P = \mathrm{tr}\left(\boldsymbol{R}_{xx}\right) \tag{式 14-2}$$

其中，$\mathrm{tr}(\boldsymbol{A})$ 代表矩阵 $\boldsymbol{A}$ 的迹，可以通过对 $\boldsymbol{A}$ 的对角元素求和得到。

如果信道状态信息（Channel State Information，CSI）在发射端未知，则假定从各个天线发射的信号都有相等的功率 P/n_T。发射信号的协方差矩阵为：

$$\boldsymbol{R}_{xx} = \frac{P}{n_T}\boldsymbol{I}_{n_T} \tag{式 14-3}$$

其中，$\boldsymbol{I}_{n_T}$ 是 $n_T \times n_T$ 的单位矩阵。

用 $n_R \times n_T$ 的复矩阵 $\boldsymbol{H}$ 描述信道。h_{ij} 为矩阵 $\boldsymbol{H}$ 的第 $i \times j$ 个元素，代表从第 j 根发射天线到第 i 根接收天线之间的信道衰落系数。用 $n_R \times 1$ 的列向量描述接收端的噪声，表示为 $\boldsymbol{n}$。它的元素是统计独立的复高斯随机变量，零均值，具有独立的、方差相等的实部和虚部。接收噪声的协方差矩阵为：

$$\boldsymbol{R}_{nn} = \sigma^2 \boldsymbol{I}_{n_R} \tag{式 14-4}$$

用 $n_R \times 1$ 的列向量描述接收信号，表示为 $\boldsymbol{y}$。使用线性模型，可将接收矢量表示成：

$$\boldsymbol{y} = \boldsymbol{Hx} + \boldsymbol{n} \tag{式 14-5}$$

接收信号的协方差矩阵定义为 $E\left\{\boldsymbol{yy}^{\mathrm{H}}\right\}$，由式 14-5 可以得出接收信号的协方差矩阵为：

$$\boldsymbol{R}_{yy} = \boldsymbol{HR}_{xx}\boldsymbol{H}^{H} + \boldsymbol{R}_{nn} \tag{式 14-6}$$

而总接收信号功率可表示为 $tr(\boldsymbol{R}_{yy})$。

14.1.2 MIMO 系统容量分析

对于给定的信道 $\boldsymbol{H}$，MIMO 系统的信道容量为发送信号向量 $\boldsymbol{x}$ 和接收信号向量 $\boldsymbol{y}$ 之间的最大互信息量。若 $\boldsymbol{x}$（它的协方差矩阵表示为 $\boldsymbol{P}$）所有元素的均值为 0，则当 $\boldsymbol{x}$ 为高斯变量时，互信息量最大，总的发射功率为 $P = tr\{\boldsymbol{P}\}$，此时信道容量为：

$$C(\boldsymbol{H}) = B\log_2\left|\boldsymbol{I} + \frac{1}{\sigma^2}\boldsymbol{HPH}^{H}\right| \tag{式 14-7}$$

其中 B 为信道带宽，$\boldsymbol{I}$ 为单位矩阵。

下面，我们就发射端是否已知 CSI，来得出 MIMO 系统的信道容量公式：

（1）如果发射端未知 CSI，按照公平原则，每个天线上分配的发射功率相同，此时 $\boldsymbol{P} = \frac{P}{n_T}\boldsymbol{I}$，信道容量为：

$$C_{UT}(\boldsymbol{H}) = B\log_2\left|\boldsymbol{I} + \frac{P}{n_T\sigma^2}\boldsymbol{HH}^{H}\right| \tag{式 14-8}$$

图 14-2 描述了发射天线分别为 2、4、8，接收天线分别为 2、4、8 时，发端未知 CSI 时的归一化信道容量。归一化信道容量是指信道容量对信道带宽的归一化，即 C/B。从图中可以看出，随着收发端天线数目的增加，归一化信道容量也随之增加。

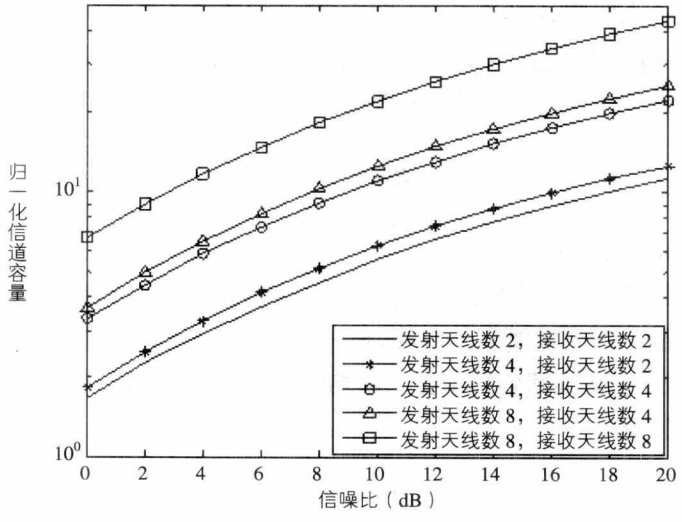


图 14-2　不同发射接收天线配置时，归一化信道容量比较

（2）如果发射端已知 CSI，那么可以根据注水原理（water-filling），通过给各个天线分配不同的发射功率，增加系统容量。其核心思想是信道条件好，则分配更多的功率；信道条件差，则分配较少的功率。

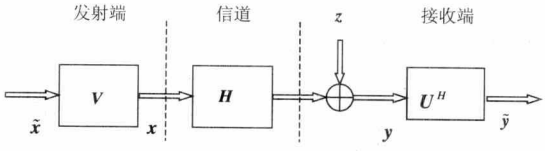


图 14-3　收发端已知 CSI，对信道进行分解

从图 14-3 中可以看出，在发射信号前先乘以一个酉阵 $\boldsymbol{V}$，在接收端，接收信号 $\boldsymbol{y}$ 乘以矩阵 $\boldsymbol{U}^H$。则输入输出关系为：

$$\begin{aligned}\tilde{\boldsymbol{y}} &= \sqrt{\frac{P}{N_t}}\boldsymbol{U}^H\boldsymbol{H}\boldsymbol{V}\tilde{\boldsymbol{x}} + \boldsymbol{U}^H\boldsymbol{z} \\ &= \sqrt{\frac{P}{N_t}}\sum\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{z}}\end{aligned} \qquad （式 14-9）$$

其中 $\tilde{\boldsymbol{y}}$ 是经过变换过后的 $r\times1$ 的接收信号，$\tilde{z}$ 是变换后的 $r\times1$ 的噪声向量，其协方差矩阵为 $E\left\{\tilde{\boldsymbol{z}}\tilde{\boldsymbol{z}}^H\right\}=\sigma^2\boldsymbol{I}_r$，而信号向量 $\tilde{\boldsymbol{x}}$ 满足 $E\left\{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^H\right\}=N_t$，保持总功率不变。式 14-10 表明，当发射机已知 CSI 时，则信道 $\boldsymbol{H}$ 可以被分解为 r 个并行 SISO 信道，即：

$$\tilde{y}_i = \sqrt{\frac{P}{N_t}}\sqrt{\lambda_i}\tilde{x}_i + \tilde{z}_i,\quad i=1,\cdots,r \qquad （式 14-10）$$

则 MIMO 信道的容量可以表示为独立并行的 SISO 信道容量之和，即：

$$C_{IT} = \sum_{i=1}^{r}\log_2\left(1+\frac{P\gamma_i}{N_t\sigma^2}\lambda_i\right) \qquad （式 14-11）$$

γ_i 是在第 i 个子信道的发射功率，并且满足总功率限制。最大化式 14-11，即得到发射机已知 CSI 时 MIMO 信道容量为：

$$C_{IT} = \max_{\sum_{i=1}^{r}\gamma_i=1}\sum_{i=1}^{r}\log_2\left(1+\frac{P\gamma_i}{N_t\sigma^2}\lambda_i\right) \qquad （式 14-12）$$

由 Lagrangian 方法得到最优的功率分配为：

$$\begin{aligned}&\gamma_i^{opt} = \left(\mu - \frac{N_t\sigma^2}{P\lambda_i}\right)_+,\quad i=1,\cdots,r \\ &\sum_{i=1}^{r}\gamma_i^{opt} = N_t\end{aligned} \qquad （式 14-13）$$

其中 μ 是常数，$(x)_+$ 表示

$$(x)_+ = \begin{cases} x, & x \geqslant 0 \\ 0, & x < 0 \end{cases} \qquad (式\ 14\text{-}14)$$

以上只是给出了注水原理的简单介绍，具体推导读者可以参考相关文献。

14.1.3 发送端信道容量的比较

发送端已知 CSI 的情况下的信道容量比发送端未知 CSI 的情况下的信道容量要高，这是由于当发送端已知 CSI 时，发送端可以优化发送信号的协方差矩阵。但是当信道的信噪比（Signal to Noise Ratio，SNR）足够大时，发送端已知 CSI 的情况下的信道容量（下面用 $C_{IT}(\boldsymbol{H})$ 表示）与发送端未知 CSI 的情况下的信道容量（下面用 $C_{UT}(\boldsymbol{H})$ 表示）相差不大。

下面给出它们二者关系的具体分析：

（1）在高 SNR 的条件下，发送端已知与未知 CSI 时信道容量的关系如下。

当 $P/\sigma^2 \to \infty$ 时，

$$\frac{C_{IT}(\boldsymbol{H})}{C_{UT}(\boldsymbol{H})} \to 1 \qquad (式\ 14\text{-}15)$$

证明： 当 $P/\sigma^2 \to \infty$ 时，

$$\begin{aligned} \frac{C_{IT}(\boldsymbol{H})}{C_{UT}(\boldsymbol{H})} &\to \frac{\log_2\left(\left|\Gamma_+\left(\frac{Tr\left\{\Gamma_+^{-2}\right\}+P}{m_-}\right)^{m_-}\right|\right)}{\log_2\left|\mathbf{I}+\frac{P}{n_t}\Gamma_+\right|} \\ &\to \frac{m_- \cdot \log_2(\frac{P}{m_-}) + \log_2\left|\Gamma_+\right|}{m_- \cdot \log_2(\frac{P}{n_t}) + \log_2\left|\Gamma_+\right|} \\ &\to 1 \end{aligned} \qquad (式\ 14\text{-}16)$$

（2）在低 SNR 的条件下，发送端已知与未知 CSI 时信道容量的关系为：

$$\frac{C_{IT}(\boldsymbol{H})}{C_{UT}(\boldsymbol{H})} \approx n_t \frac{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^H)}{Tr\{\boldsymbol{H}\boldsymbol{H}^H\}} \qquad (式\ 14\text{-}17)$$

证明： 在低 SNR 的条件下，MIMO 系统信道可以近似等效为信道增益为 $\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^H)$ 的 SISO 系统，所以有

$$\begin{aligned} \frac{C_{IT}(\boldsymbol{H})}{C_{UT}(\boldsymbol{H})} &\to \frac{\log_2\left(1+\frac{P}{\sigma^2}\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^H)\right)}{\log_2\left|\boldsymbol{I}+\frac{P}{n_t\sigma^2}\boldsymbol{H}\boldsymbol{H}^H\right|} \\ &\approx n_t \frac{\lambda_{\max}(\boldsymbol{H}\boldsymbol{H}^H)}{Tr\{\boldsymbol{H}\boldsymbol{H}^H\}} \end{aligned} \qquad (式\ 14\text{-}18)$$

在 SNR 足够大的条件下，$C_{IT}(\boldsymbol{H})$ 虽然可以收敛于 $C_{UT}(\boldsymbol{H})$，但是在实际系统中 SNR

总是受到约束的，不可能无止境地增加。当增加通信系统的工作 SNR 时，不仅对发射机功率放大器提出了更高的要求，同时也增加了接收机的功耗，这与终端系统节电的原则不符合。因此在实际系统中，SNR 不可能满足足够大的条件。根据以上分析可知，注水原理无疑是信道容量达到最大的最佳选择，在实际系统中，发射端必须有效利用 CSI，这样可以优化发送信号。

图 14-4 比较了不同收发天线配置下，发送端已知与未知 CSI 时的归一化信道容量。与前面的分析一致，从该图中可以看出，随着信噪比的不断增大，$C_{IT}(\boldsymbol{H})$ 逐渐收敛于 $C_{UT}(\boldsymbol{H})$。还有一点值得注意的是，当收发天线数目相等时，已知 CSI 时的归一化信道容量仅略高于未知 CSI 时；当发射天线数大于接收天线数时，已知 CSI 时的归一化信道容量要明显高于未知 CSI 时。实际中由于移动终端受各种因素的限制，它与基站端的天线数目是不平衡的，一般来说，移动终端所支持的天线数目总是比基站端少。

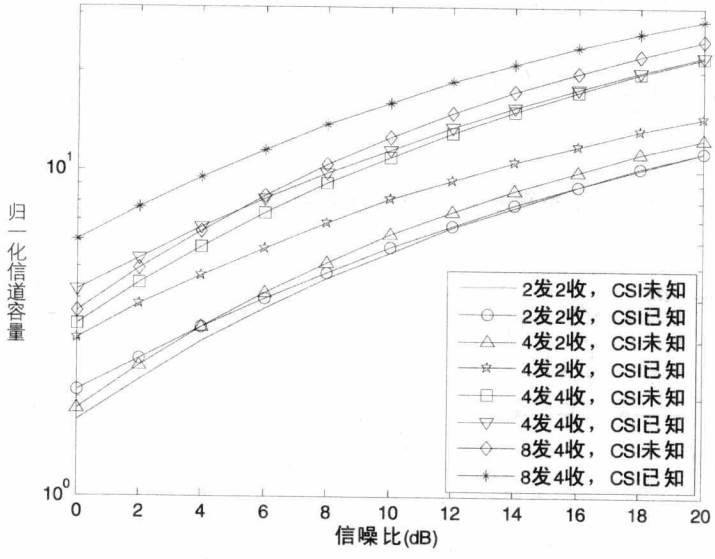


图 14-4 不同收发天线配置时，已知与未知 CSI 时归一化信道容量比较

14.2 OFDM 技术简介

OFDM 的思想早在 20 世纪 60 年代就已经被提出，但由于使用模拟滤波器实现起来的系统复杂度较高，所以一直没有发展起来。在 20 世纪 70 年代，S.B. Weinstein 提出用离散傅里叶变换实现多载波调制，为 OFDM 的实用化奠定了理论基础。20 世纪 80 年代，L.J.Cimini 分析了 OFDM 在移动通信应用中存在的问题和解决方法，从此以后，OFDM 技术在移动通信中的应用得到了迅猛的发展。在前面一章中对 OFDM 系统已经做了详细的讨论，这里对 OFDM 的基本原理仅进行简单介绍。

OFDM 的基本原理是把高速的数据流通过串并变换，分配到传输速率相对较低的若干个子信道中进行并行传输。由于每个子信道中的符号周期会相对增加，因此可以减轻无线信道的多径时延扩展对系统造成的影响。通过在 OFDM 符号之间插入保护间隔，并且令保护

间隔大于无线信道的最大时延扩展，就可以最大限度地消除符号间干扰（ISI）。而且，一般都采用循环前缀作为保护间隔，从而可以避免子载波间干扰（Inter-Carrier Interference，ICI）。

一个 OFDM 符号由多个经过调制的子载波信号合成，其中每个子载波可以采用相移键控（Phase Shift Keying，PSK）或者正交幅度调制（Quadrature Amplitude Modulation，QAM）符号的调制。如果 N 表示子载波的个数，T 表示 OFDM 符号的宽度，$d_i(i=0,1,\cdots,N-1)$ 是分配给每个子载波的数据符号，f_c 是第 0 个子载波的载波频率，$rect(t)=1,|t|\leqslant T/2$，则从 $t=t_s$ 开始的 OFDM 符号可以表示为：

$$s(t)=\begin{cases}\mathrm{Re}\left\{\sum_{i=0}^{N-1}d_i rect(t-t_s-\frac{T}{2})\exp\left[\mathrm{j}2\pi(f_c+\frac{i}{T})(t-t_s)\right]\right\} & t_s\leqslant t\leqslant t_s+T\\ 0 & t<t_s \quad \wedge \quad t>t+t_s\end{cases} \tag{式 14-19}$$

采用复等效基带信号来描述 OFDM 的输出信号，如式 14-20 所示。

$$s(t)=\begin{cases}\sum_{i=0}^{N-1}d_i rect(t-t_s-\frac{T}{2})\exp\left[\mathrm{j}2\pi\frac{i}{T}(t-t_s)\right] & t_s\leqslant t\leqslant t_s+T\\ 0 & t<t_s \quad \wedge \quad t>t+t_s\end{cases} \tag{式 14-20}$$

图 14-5 给出了 OFDM 系统基本模型框图，其中 $f_i=f_c+i/T$。

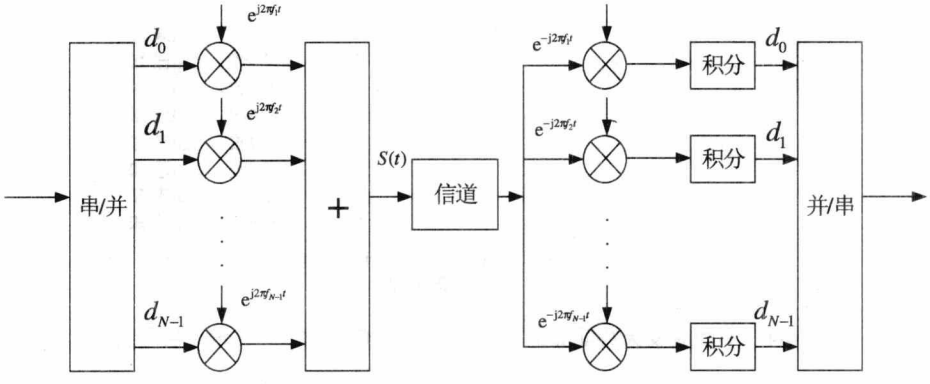


图 14-5　OFDM 系统基本模型框图

每个子载波在一个 OFDM 符号周期内都包含整数倍周期，而且各个相邻的子载波之间相差 1 个周期。这一特性可以用来解释子载波间的正交性，即：

$$\frac{1}{T}\int_0^T \mathrm{e}^{\mathrm{j}\omega_n t}\cdot\mathrm{e}^{-\mathrm{j}\omega_n t}\mathrm{d}t=\begin{cases}1, & n=m\\ 0, & n\neq m\end{cases} \tag{式 14-21}$$

对式 14-20 中的第 j 个子载波进行解调，然后在时间长度 T 内进行积分，即：

$$\begin{aligned}\hat{d}_j&=\frac{1}{T}\int_{t_s}^{t_s+T}\exp\left(-\mathrm{j}2\pi\frac{j}{T}(t-t_s)\right)\sum_{i=0}^{N-1}d_i\exp\left(\mathrm{j}2\pi\frac{i}{T}(t-t_s)\right)\mathrm{d}t\\&=\frac{1}{T}\sum_{i=0}^{N-1}d_i\int_{t_s}^{t_s+T}\exp\left(\mathrm{j}2\pi\frac{i-j}{T}(t-t_s)\right)\mathrm{d}t=d_j\end{aligned} \tag{式 14-22}$$

根据式 14-22 可以看到，对第 j 个子载波进行解调可以恢复出期望符号。而对其他载波来说，由于在积分间隔内，频率相差 $(i-j)/T$ 可以产生整数倍个周期，所以积分结果为零。

当 N 很大时，需要大量的正弦波发生器、滤波器、调制器和解调器等设备，因此系统非常昂贵。为了降低 OFDM 系统的复杂度和成本，通常考虑用离散傅里叶变换（Discrete Fourier Transform，DFT）和离散傅里叶逆变换（Inverse Discrete Fourier Transform，IDFT）来实现上述功能。对式 14-20 中等效复基带信号以 T/N 的速率进行抽样，即令 $t=kT/N$ $(k=0,1,\cdots,N-1)$，则可得到：

$$s_k = s(kT/N) = \sum_{i=0}^{N-1} d_i \exp\left(\mathrm{j}\frac{2\pi ik}{N}\right) \qquad (0 \leqslant k \leqslant N-1) \qquad （式 14\text{-}23）$$

可见 s_k 即是对 d_i 进行 IDFT 运算，容易推得在接收端同样可以用 DFT 恢复原始的数据信号，在接收端对接收到的 s_k 进行 DFT 变换即得：

$$d_i = \sum_{k=0}^{N-1} s_k \exp\left(-\mathrm{j}\frac{2\pi ik}{N}\right) \qquad (0 \leqslant i \leqslant N-1) \qquad （式 14\text{-}24）$$

在 OFDM 系统的实际运用中，可以采用更加方便快捷的 IFFT/FFT。N 点 IDFT 运算需要实施 N^2 次的复数乘法，而 IFFT 可以显著地降低运算的复杂度。

14.3 MIMO-OFDM 系统结构

利用 MIMO 技术和 OFDM 技术两者各自的特点，结合形成的 MIMO-OFDM 系统，将空间分集、时间分集以及频率分集有机地结合起来，从而能够大大地提高无线通信系统的信道容量和传输速率，有效抵抗信道衰落和抑制干扰，被业界认为是构建未来宽带无线通信系统最关键的物理层传输方案。

如图 14-6 所示，在 MIMO-OFDM 系统中，每根发射天线的通路上都有一个 OFDM 调制器，每根接收天线的通路上也都有一个 OFDM 的解调器。

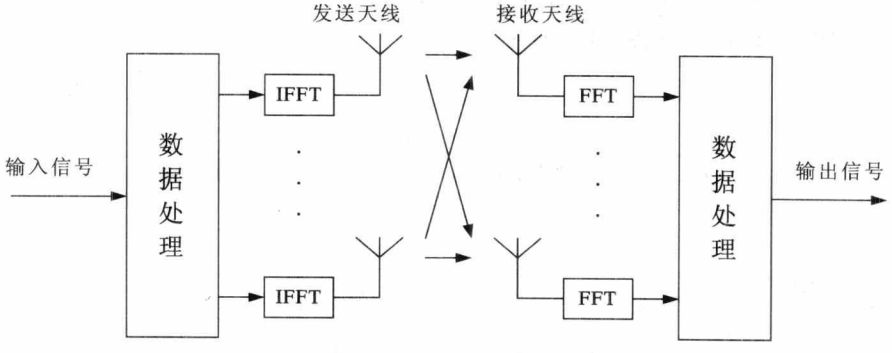


图 14-6　$N_t \times N_r$ MIMO-OFDM 系统结构框图

由于 OFDM 技术能够将频率选择性衰落信道转化为若干个平坦衰落的并行子信道，因此 MIMO-OFDM 系统中任意一个子载波上的输入输出关系相当于一个平坦衰落信道

MIMO 系统，可以表示为：

$$\boldsymbol{y}_k[t]=\boldsymbol{H}_k[t]\boldsymbol{x}_k[t]+\boldsymbol{n}_k[t] \qquad (式\ 14\text{-}25)$$

其中，$\boldsymbol{y}_k[t]$ 表示第 t 个时隙（这里一个时隙指一个 OFDM 符号），第 k 个 OFDM 子载波上 $N_r\times1$ 的接收符号向量；N_r 为接收天线数目；$\boldsymbol{x}_k[t]$ 表示第 t 个时隙，第 k 个子载波上 $N_t\times1$ 的发射符号向量；N_t 为发射天线数目；$\boldsymbol{H}_k[t]$ 表示第 t 个时隙，第 k 个子载波上 $N_t\times N_r$ 的 MIMO 复信道系数矩阵，这里假定信道系数在每个 OFDM 符号周期内保持不变；$\boldsymbol{n}_k[t]$ 表示第 t 个时隙，第 k 个子载波上 $N_r\times1$ 的接收天线上复高斯噪声向量，其每个元素的均值为 0，方差为 σ^2。这里，向量 $\boldsymbol{n}_k[t]$ 满足 $E\left\{\boldsymbol{n}_k[t]\boldsymbol{n}_k[t]^H\right\}=\sigma^2\boldsymbol{I}_{N_r}$，$\boldsymbol{I}_{N_r}$ 表示 $N_t\times N_r$ 的单位阵，$E\{\cdot\}$ 表示数学期望，$\boldsymbol{n}_k[t]^H$ 表示 $\boldsymbol{n}_k[t]$ 的共扼转置。

14.4 空时编码技术

对 MIMO 系统性能的研究表明：在假设各天线相互独立的条件下，多天线系统比单天线系统在信道容量上有显著提高，这些增加的信道容量既可以用来提高信息传输速率，也可以通过增加信息冗余来提高通信系统的传输可靠性。能够获得 MIMO 系统这些好处的一种行之有效的方法是进行空时编码：该方法联合考虑信道编码、调制、发送和接收分集，将它们有机结合，有效地提高了 MIMO 系统的传输性能。目前研究较多的空时编码有基于发送分集的分层空时编码（BLAST）、空时网格编码（STTC）和空时分组编码（STBC）。

14.4.1 分层空时编码（BLAST）

分层空时码（BLAST）首先是由 Bell 实验室 Foschini 等人提出的，它将信源数据分为若干子数据流，独立地进行编码、调制，因而它不是基于发送分集的。它的基本思想是把高速的数据业务分解为若干低速数据业务，通过普通的信道编码器编码后，对其进行分层次的空时编码，调制后再经过多副天线发送。在接收端，用多副天线进行分集接收。由于分层空时码的译码需要用到信道传输特性，所以需要进行信道估计，然后进行分层空时译码。其收发模型如图 14-7 和图 14-8 所示。

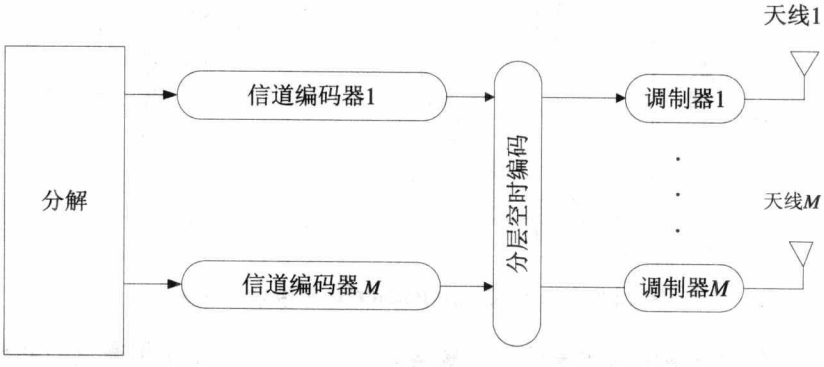


图 14-7 分层空时码发送模型

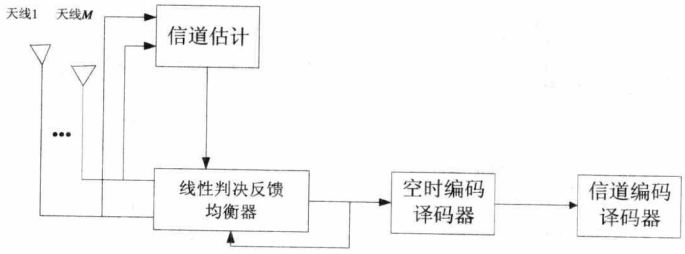


图 14-8　分层空时码接收模型

从分层空时码的接收模型可以看出每副天线上的接收信号是不同发送天线信号的叠加，这有点类似于多址接入系统。在多址接入系统中，每个用户接收的信号除了有用信号外还有来自其他用户的多址干扰，需要消除多址干扰后得到有用信号；分层空时码需要消除其他天线上信号的干扰。但是两者之间也有区别，多址接入研究的是多个用户间的通信，而分层空时编码则是单个用户的情况。在多址接入系统中，通过给不同的用户分配不同的资源，且满足正交关系，接收端可以利用这种正交关系来消除多址干扰。而分层空时编码则是在同一频率和时间上发送同一用户的信息，因此它的传输效率要高得多。

假设发送天线数目为 n_t，接收天线数目为 n_r，$h_{j,i}$ 表示的是发送天线 i（$i=1,2,\cdots,\ n_t$）到接收天线 j（$j=1,2,\cdots,\ n_r$）之间的信道衰落系数，$h_{j,i}$ 是服从均值为 0、方差为 1 的高斯随机变量，以所有 $h_{j,i}$ 为元素构成信道矩阵 $\boldsymbol{H}\in\boldsymbol{C}^{n_r\times n_t}$，信道满足平坦瑞利快衰落模型。第 j 副接收天线收到的信号记为 r_j（$j=1,2,...,n_r$），记列向量：

$$\boldsymbol{r}=\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n_1} \end{bmatrix} \qquad \boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_2} \end{bmatrix} \qquad \boldsymbol{N}=\begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_{n_3} \end{bmatrix} \tag{式 14-26}$$

分别为接收信号向量 $\boldsymbol{r}$、发送信号向量 $\boldsymbol{X}$ 和噪声向量 $\boldsymbol{N}$，$N_j\left(j=1,2,\cdots,n_r\right)$ 为每副接收天线的加性高斯白噪声（AWGN），则有

$$\boldsymbol{r}=\boldsymbol{HX}+\boldsymbol{N} \tag{式 14-27}$$

研究表明，分层空时编码的优点是当接收天线数 n_r 大于发送天线数 n_t 时，其系统容量是与发送天线数 n_t 成正比增长的：

$$\boldsymbol{C}=n_t\log_2\left(\frac{n_r}{n_t}\rho\right),\quad n_r\geqslant n_t \tag{式 14-28}$$

其中，ρ 为每副接收天线的平均接收信噪比。

14.4.2　空时网格编码（STTC）

Tarokh 等人在空时延迟分集（Delay diversity）、Alamouti 关于发送分集等研究工作的基础上，将这种发送分集技术结合网格编码调制（TCM）技术提出了空时网格编码（STTC，

Space-Time Trellis Coding）技术。采用两副发送天线的空时网格编码。系统的传输性能与 Foschini 等人给出的中断容量仅相差 2～3dB。空时网格编码的系统如图 14-9 所示。

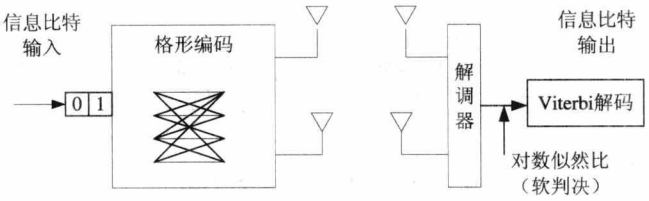


图 14-9　空时网格编码系统图

空时格栅编码原理可以用格栅图来表示。以 QPSK 调制 4 状态空时格形码为例，假定为两根发射天线。图 14-10 画出了 STTC 的 QPSK 星座图和格栅图表示。令 b_1b_2 分别表示从两个天线发射出去的字符。

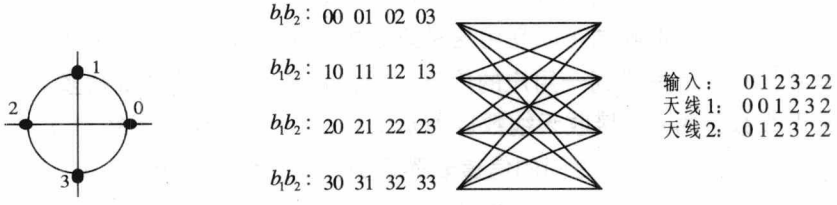


图 14-10　格栅码编码示意图

具体编码过程如下：设编码器初始状态为 0，输入字符为（01,10,11,10,10）。由于编码器初始状态为 0，发送的第一个比特为 1，从格栅图看出，这对应于格栅图中从状态 0 出发的第 2 条线，下一时刻的状态转移到了 1，编码器的输出是左边的 01，也就是说，天线 1 发送 0，天线 2 发送 1；发射的第二个比特为 2，这对应于格栅图中从状态 1 出发的第三条线，编码器的输出是左边的 12，即天线 1 发送 1，天线 2 发送 2；发射的第三个比特为 3，这对应于格栅图中从状态 2 出发的第四条线，编码器的输出是左边的 23，即天线 1 发送 2，天线 2 发送 3，依次类推，可以得到（01,12,23,32,22）。空时格形码编码与卷积编码一样，需要在信息序列后面加上若干个 0 比特信息，使得编码器的最后状态回到 0 状态，这样做是为了以后 Viterbi 译码的方便。

天线 1 上的发送符号为：（0 1 2 3 2）

天线 2 上的发送符号为：（1 2 3 2 2）

STTC 建立了空域（发送天线）中信号和时域（连续时间）中信号的内在联系，提供最大可能的分集增益和编码增益，不会牺牲发射带宽。接收机是基于信道衰落系数估计和 ML（Maximum Likelihood）的 Viterbi 译码或者是最大后验概率 MAP 译码，当天线数目固定时，译码复杂度随发射速率呈指数增加，显得不太实用。

14.4.3　空时分组编码（STBC）

为了克服空时格栅译码过于复杂的缺陷，Alamouti 于 1998 年发明了使用两个天线发

射的空时分组编码（STBC）。

Alamouti 提出的简单发送分集方案如图 14-11 所示。信源发送的二进制信息比特首先进行调制（星座映射）。假设采用 M 进制的调制星座，有 $m=\log_2 M$。把从信源来的二进制信息比特每 m 个比特分为一组，对连续的两组比特进行星座映射，得到两个调制符号 x_1 与 x_2。然后把这两个符号送入编码器，并按照下面的方式编码：

$$\begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} \qquad （式 14\text{-}29）$$

经过编码后的符号分别从两副天线上发送出去；在第一个发送时刻，符号 x_1 与 x_2 发送天线 1 和发送天线 2 上同时发送出去；在第二个发送时刻，符号 $-x_2^*$ 与 x_1^* 分别从发送天线 1 和发送天线 2 上同时发送出去，如图 14-11 所示。

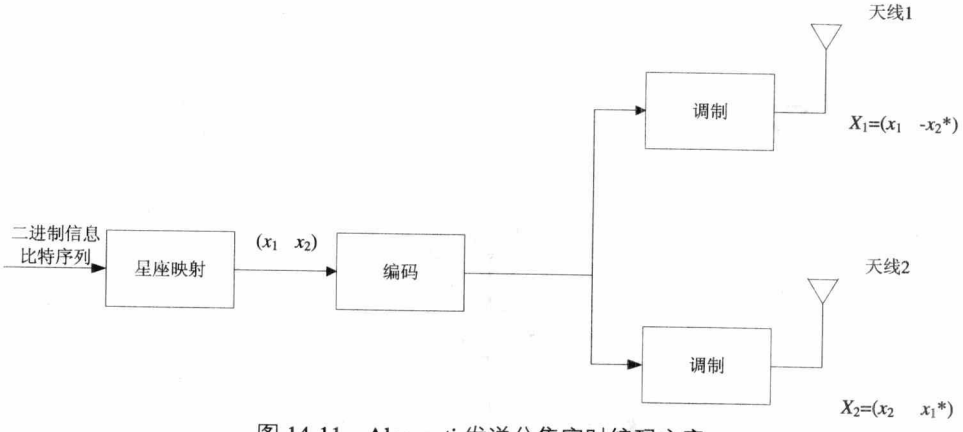


图 14-11　Alamouti 发送分集空时编码方案

从编码过程中可以看出，由于在时间和空间域同时进行了编码，故名为空时编码，从两副发送天线上发送信号批次存在着一定的关系，因此这种空时码是基于发送分集的。式 14-29 的编码矩阵满足：

$$XX^* = \begin{bmatrix} |x_1|^2+|x_2|^2 & 0 \\ 0 & |x_1|^2+|x_2|^2 \end{bmatrix} = \left(|x_1|^2+|x_2|^2\right) I_2 \qquad （式 14\text{-}30）$$

故其是满足列正交的，即同一符号内，从两副发送天线上发送的信号满足正交性。记 X_1 和 X_2 分别为从发送天线 1 和发送天线 2 上发送的符号，则有

$$\begin{aligned} X_1 &= (x_1 \quad -x_2^*) \\ X_2 &= (x_2 \quad x_1^*) \\ X_1 X_2^* &= x_1 x_2^* - x_1 x_2^* = 0 \end{aligned} \qquad （式 14\text{-}31）$$

空时分组码也正式由于满足式 14-30 和式 14-31 的正交性才使得译码相对简单，这一点可以从后面的译码方法中看出。

图 14-12 是在接收端有一副接收天线时 Alamouti 空时码的接收机。假设在时刻 t 发送天线 1 和发送天线 2 到接收天线的信道衰落系数分别为 $h_1(t)$ 与 $h_2(t)$，再考虑到快衰落信道假设，有

$$\begin{aligned} h_1(t) &= h_1(t+T) = h_1 = |h_1| e^{j\theta} \\ h_2(t) &= h_2(t+T) = h_2 = |h_2| e^{j\theta} \end{aligned} \quad (式\ 14\text{-}32)$$

$|h_i|$ 和 $\theta_i\ (i=1,2)$ 是发送天线 i 到接收天线信道的幅度响应与相位偏转，T 表示符号间隔。记接收天线在时刻 t 和 $t+T$ 的接收信号分别为 r_1 与 r_2，有

$$\begin{aligned} r_1 &= h_1 x_1 + h_2 x_2 + n_1 \\ r_2 &= -h_1 x_2^* + h_2 x_1^* + n_2 \end{aligned} \quad (式\ 14\text{-}33)$$

n_1与n_2表示接收天线在时刻 t 和 $t+T$ 的独立复高斯白噪声，假设噪声的均值为 0，每维的方差为 $N_0/2$。

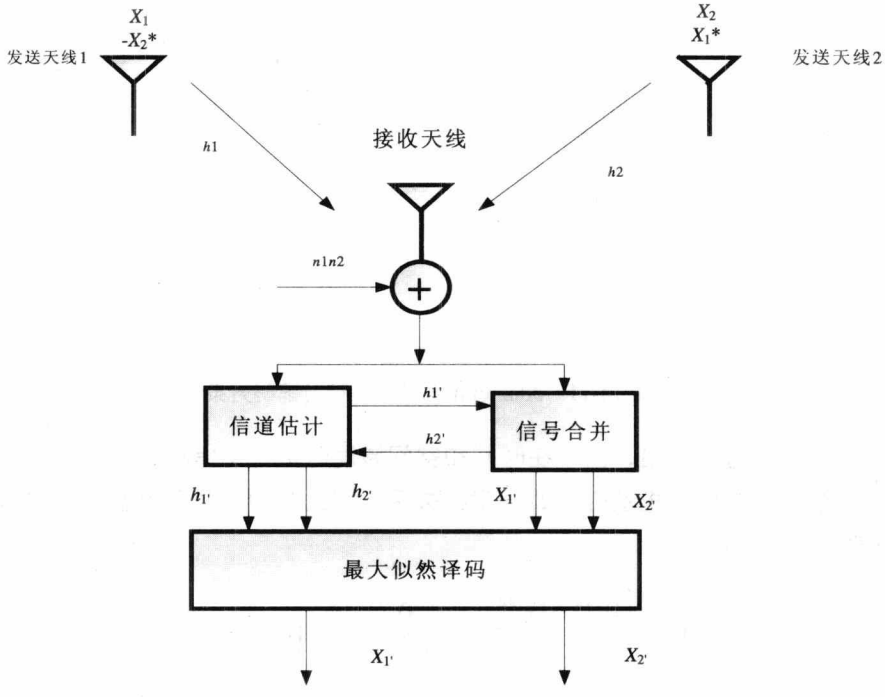


图 14-12　Alamouti 空时码的接收机

假设接收端可以完全准确地估计出信道的衰落系数 h_1 和 h_2，在接收端采用最大似然译码，即从星座中找出一对符号$(\hat{x}_1, \hat{x}_2)$，使下式的欧式距离最小：

$$\begin{aligned} & d^2\left(r_1, h_1\hat{x}_1 + h_2\hat{x}_2\right) + d^2\left(r_2, -h_1\hat{x}_2^* + h_2\hat{x}_1^*\right) \\ &= \left|r_1 - h_1\hat{x}_1 - h_2\hat{x}_2\right|^2 + \left|r_2 + h_1\hat{x}_2^* - h_2\hat{x}_1^*\right|^2 \rightarrow \min \end{aligned} \quad (式\ 14\text{-}34)$$

这里假设星座中的符号等概率发送。把式 14-33 代入到式 14-34 中，最大似然译码

变成：

$$\begin{aligned}(\hat{x}_1,\hat{x}_2)=&\arg\min\left(\left(|h_1|^2+|h_2|^2-1\right)\left(|\hat{x}_1|^2+|\hat{x}_2|^2\right)\right.\\&\left.+d^2(\tilde{x}_1,\hat{x}_1)+d^2(\tilde{x}_2,\hat{x}_2),\quad (\hat{x}_1,\hat{x}_2)\in C\right.\end{aligned}\tag{式 14-35}$$

其中，C 是所有可能发送符号的集合，$\tilde{x}_1$与$\tilde{x}_2$ 是根据信道衰落系数和接收信号进行合并得到的信号。

$$\begin{aligned}\tilde{x}_1&=h_1^*r_1+h_2r_2^*=\left(|h_1|^2+|h_2|^2\right)x_1+h_1^*n_1+h_2n_2^*\\\tilde{x}_2&=h_2^*r_1-h_1r_2^*=\left(|h_1|^2+|h_2|^2\right)x_2-h_1n_2^*+h_2^*n_1\end{aligned}\tag{式 14-36}$$

从式 14-36 中可以看出在接收端已经获得信道衰落系数 h_1与h_2 的情况下，合并信号 $\tilde{x}_1$与$\tilde{x}_2$ 分别是 x_1与x_2 的函数。由于 $\tilde{x}_1$、$\tilde{x}_2$ 中没有 x_1、x_2 的交叉项，故根据式 14-35 可分解出两个独立信号并分别对其进行译码，可得

$$\begin{aligned}\hat{x}_1&=\arg\min\left(|h_1|^2+|h_2|^2-1\right)|\hat{x}_1|^2+d^2(\tilde{x}_1,\hat{x}_1),\hat{x}_1\in S\\\hat{x}_2&=\arg\min\left(|h_1|^2+|h_2|^2-1\right)|\hat{x}_2|^2+d^2(\tilde{x}_2,\hat{x}_2),\hat{x}_2\in S\end{aligned}\tag{式 14-37}$$

其中，S 为调制映射星座。若采用 MPSK 星座，所有星座点的功率相等，则判决式 14-37 可以简化为

$$\begin{aligned}\hat{x}_1&=\arg\min d^2(\tilde{x}_1,\hat{x}_1),\hat{x}_1\in S\\\hat{x}_2&=\arg\min d^2(\tilde{x}_2,\hat{x}_2),\hat{x}_2\in S\end{aligned}\tag{式 14-38}$$

式 14-38 是 Alamouti 空时码在单副接收天线下采用 MPSK 调制进行最大似然译码时的判决度量。

Alamouti 提出的发送分集方案也可以扩展到两副发送天线和多副接收天线系统中。在多接收天线系统中，发送端的编码与传输方案与单接收天线系统是一样的，此处不再赘述。但是在接收端的处理变得复杂，需要对不同接收天线上接收到的信号进行合并处理。仍然采用式 14-29 的 Alamouti 空时编码方案，记 r_1^j与r_2^j 分别表示第 j 副天线在时刻 t 与 $t+T$ 接收到的信号，有

$$\begin{aligned}r_1^j&=h_{j,1}x_1+h_{j,2}x_2+n_1^j\\r_2^j&=-h_{j,1}x_2^*+h_{j,2}x_1^*+n_2^j\end{aligned}\tag{式 14-39}$$

其中 $h_{j,i}\,(i=1,2;j=1,2,\cdots,n_r)$ 表示发送天线 i 到接收天线 j 的信道衰落系数，n_1^j与n_2^j 分别表示接收天线 j 在时刻 t 和 $t+T$ 时的噪声信号。

单副接收天线下的判决度量可以通过式 14-36 得到，多接收天线下的判决度量可以通过把各副接收天线上的接收信号根据式 14-36 所得到的判决度量进行线性合并得到。

$$\overline{x}_1=\sum_{j=1}^{n_r}\left[h_{j,1}^{*}r_1^j+h_{j,1}\left(r_2^j\right)^{*}\right]=\sum_{i=1}^{2}\sum_{j=1}^{n_r}\left|h_{j,i}\right|^2x_1+\sum_{j=1}^{n_r}h_{j,1}^{*}n_1^j+h_{j,2}\left(r_2^j\right)^{*} \quad (式\ 14\text{-}40)$$

$$\overline{x}_2=\sum_{j=1}^{n_r}\left[h_{j,2}^{*}r_1^j-h_{j,1}\left(r_2^j\right)^{*}\right]=\sum_{i=1}^{2}\sum_{j=1}^{n_r}\left|h_{j,i}\right|^2x_2+\sum_{j=1}^{n_r}h_{j,2}^{*}n_1^j-h_{j,1}\left(n_2^j\right)^{*} \quad (式\ 14\text{-}41)$$

同理，根据式 14-37 可以得到

$$\hat{x}_1=\arg\min\left\{\left[\sum_{j=1}^{n_r}\left(\left|h_{j,1}\right|^2+\left|h_{j,2}\right|^2\right)-1\right]\left|\hat{x}_1\right|^2+d^2\left(\overline{x}_1,\hat{x}_1\right)\right\},\hat{x}_1\in S \quad (式\ 14\text{-}42)$$

$$\hat{x}_2=\arg\min\left\{\left[\sum_{j=1}^{n_r}\left(\left|h_{j,1}\right|^2+\left|h_{j,2}\right|^2\right)-1\right]\left|\hat{x}_2\right|^2+d^2\left(\overline{x}_2,\hat{x}_2\right)\right\},\hat{x}_2\in S \quad (式\ 14\text{-}43)$$

14.5 基于 STBC 的 MIMO-OFDM 系统设计

现有的空时编码理论大都基于平坦衰落信道，而实际上大多数无线通信环境都属于频率选择性衰落信道，此时系统的性能会急剧下降，这就使得空时编码在宽带移动通信中的应用受到极大的限制。将 STBC 与 OFDM 相结合得到的 STBC-OFDM 系统实际上就是将空时分组编码应用在频率选择性衰落信道的一种方案，一方面空时分组编码具有相对简单的编译码算法和较好的性能，另一方面 OFDM 能有效地对抗多径衰落，将频率选择性衰落信道转换为并行的平坦衰落信道，因此 STBC-OFDM 系统可以提供更高的传输速率和频谱效率，具有更好的性能。

下面着重讨论空时编码技术与 OFDM 技术的结合，对其性能进行详细分析，并给出基于 STBC 的 MIMO-OFDM 系统设计。

14.5.1 STBC-MIMO-OFDM 系统模型

有 N 副发射天线、M 副接收天线的 STBC-OFDM 系统框图如图 14-13 所示，信号经过 MIMO 频率选择性衰落信道。设系统总带宽被划分为 K 个相互重叠的子信道。每个空时码字包含 NK 个码符号，在一个 OFDM 码字持续时间内同时发送，每个码符号用某一发射天线在某一 OFDM 的子载波上发送，假定衰落是准静态的，即在 OFDM 的一帧内衰落保持不变且不同的发射天线和接收天线对之间的衰落是不相关的。

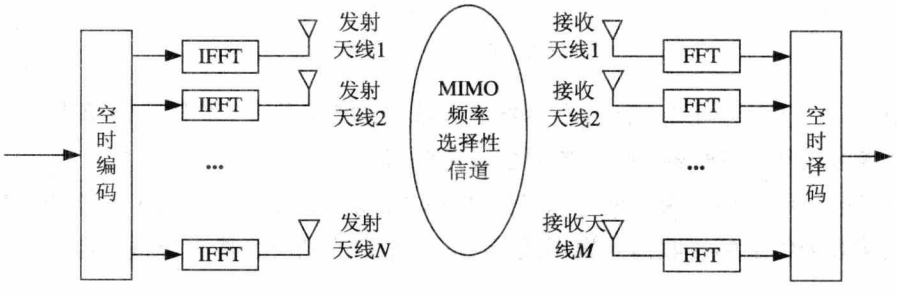


图 14-13 （N，M）STBC-MIMO-OFDM 系统框图

如图 14-13 所示，在每一时刻 t，空时编码器对输入的信息比特序列进行编码输出码字：

$$c^t = c_{10}^t, c_{20}^t, ..., c_{N0}^t, c_{11}^t, c_{21}^t, ..., c_{N1}^t, ..., c_{1(K-1)}^t, c_{2(K-1)}^t, ..., c_{N(K-1)}^t \quad (式\ 14\text{-}44)$$

其中 c_{ik}^t 为时刻 t 在第 i 副发射天线的第 k 个子载波上传输的数据，是调制星座中的星座点信号。然后分别对信号点序列：

$$c_{i0}^t, c_{i1}^t, ..., c_{i(K-1)}^t,\ \ i = 1, 2, \cdots, N$$

进行 OFDM 调制，并映射到第 i 副发射天线上（$i = 1, 2, \cdots, N$），最后将这些调制信号由 N 副发射天线在时刻 t 同时发射出去。

为了消除由于信道时延扩展而引起的码间干扰 ISI，OFDM 系统中通常引入循环前缀，假定循环前缀长度大于信道最大时延扩展，且系统收发端完全同步，那么，接收天线 j（$j = 1, 2, \cdots, M$）上的接收信号经符号速率采样，去循环前缀及 FFT，解调后为：

$$R_{jk}^t = \sum_{i=1}^{N} H_{ijk}^t c_{ik}^t + N_{jk}^t \quad (k = 0, 1, \cdots, K-1) \quad (式\ 14\text{-}45)$$

其中 H_{ijk}^t 为 t 时刻从第 i 副发射天线到第 j 副接收天线之间的信道在第 k 个子载波频率处的频率响应，N_{jk}^t 表示接收端噪声和干扰的复高斯随机变量。

假设接收端已知信道状态信息，则收端译码器运用最大似然检测算法，寻找使度量值

$$\sum_{j=1}^{M} \sum_{k=0}^{K-1} \left| R_{jk}^t - \sum_{i=1}^{N} H_{ijk}^t Q_{ik} \right|^2 \quad (式\ 14\text{-}46)$$

最小的码字

$$Q = Q_{10}, Q_{20}, ..., Q_{N0}, Q_{11}, Q_{21}, ..., Q_{N1}, ..., Q_{1(K-1)}, Q_{2(K-1)}, ..., Q_{N(K-1)} \quad (式\ 14\text{-}47)$$

作为译码器的输出。

14.5.2 STBC-MIMO-OFDM 系统性能分析

为分析简单起见，把接收信号式 14-45 表示为矩阵形式：

$$Y[k] = H[k]X[k] + Z[k],\ \ k = 0, 1, \cdots, K \quad (式\ 14\text{-}48)$$

其中 $H[k] \in C^{M \times N}$ 为第 k 个子载波处的复信道频率响应矩阵，$X[k] \in C^N$ 和 $Y[k] \in C^M$ 分别为第 k 个子载波上的发射信号和接收信号，$Z[k] \in C^M$ 为加性噪声，设其为具有单位方差的复高斯随机变量。

下面讨论第 j 副发射天线与第 i 副接收天线之间的信道响应。其时域脉冲响应用抽头延时线模拟（仅考虑非零的抽头）可表示为：

$$h_{ij}(\tau;t)=\sum_{l=1}^{L}a_{ij}(l;t)\delta\left(\tau-\frac{n_l}{K\Delta f}\right) \quad (式\ 14\text{-}49)$$

其中 $\delta(\cdot)$ 为冲击函数；L 为非零抽头的个数；$a_{ij}(l;t)$ 为第 l 个非零抽头的复幅值，其延时为 $\frac{n_l}{K\Delta f}$，n_l 为一整数；Δf 为 OFDM 系统的各子载波之间的频率间隔。由于已假设信道为准静态的，即在 OFDM 的一帧内衰落保持不变，且在此仅讨论在 OFDM 的一帧内空时码的发射和接收，因此为表示方便可将表示时间的参数 t 省略。

由式 14-49 第 j 副发射天线与第 i 副接收天线之间的信道在第 k 个子载波处的频率响应，也就是式 14-48 中 $H[k]$ 的第 i 行第 j 列的元素为：

$$H_{ij}[k]=H_{ij}[k\Delta f]=\sum_{l=1}^{L}h_{ij}[l]\mathrm{e}^{-\mathrm{j}2\pi kn_l/k}=h_{ij}^{*}w_f(k) \quad (式\ 14\text{-}50)$$

其中，$h_{ij}[l]=a_{ij}(l)$，$h_{ij}=[a_{ij}(1),a_{ij}(2),\cdots,a_{ij}(L)]^{*}$ 为包含所有非零抽头的时域频率响应的 L 维向量，$w_f(k)=\left[\mathrm{e}^{-\mathrm{j}2\pi kn_1/K},\mathrm{e}^{-\mathrm{j}2\pi kn_2/K},\cdots,\mathrm{e}^{-\mathrm{j}2\pi kn_L/K}\right]^{T}$ 则包含相应的离散傅里叶变换的系数。

假设接收端已知信道状态信息，则与式 14-48 相应的最大似然 ML 检测度量值为：

$$\sum_{i=1}^{M}\sum_{k=0}^{K-1}\left|y_i[k]-\sum_{i=1}^{N}H_{ij}[k]x_j[k]\right|^2 \quad (式\ 14\text{-}51)$$

使其最小的码字

$$x=x_1[0],x_1[1],\cdots,x_1[K-1],x_2[0],x_2[1],\cdots,x_2[K-1],x_N[0],x_N[1],\cdots,x_N[K-1] \quad (式\ 14\text{-}52)$$

即为译码器的输出。

14.5.3 STBC-MIMO-OFDM 通信系统设计

1. 系统仿真参数

根据前面已经介绍的 STBC-MIMO-OFDM 通信系统的基本结构，表 14-1 给出了仿真系统的主要参数。

表 14-1 STBC-MIMO-OFDM 通信系统仿真

参　数	参数值
子载波数	100
IFFT/FFT	512
循环前缀	10
调制方式	QPSK
天线设置	发射天线数：2 或 3；接收天线数：2
信道	4 多径和高斯白噪声

2. 系统性能仿真

STBC-MIMO-OFDM 通信系统性能仿真如图 14-14 所示（发射天线数为 3，接收天线数为 2）。两张图分别是两个接收天线的误码率仿真曲线。

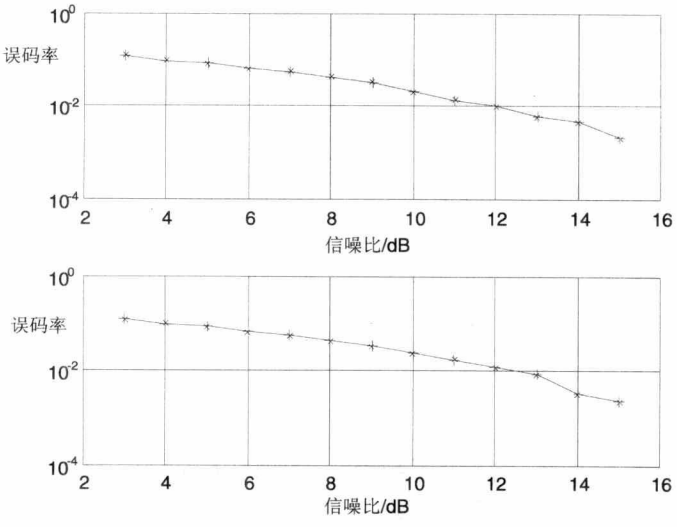


图 14-14　STBC-MIMO-OFDM 通信系统性能

14.6 基于 STBC 的 MIMO-OFDM 通信系统仿真程序

下面列出基于 STBC 的 MIMO-OFDM 通信系统的 MATLAB 仿真程序与注释，便于读者借鉴参考。

```
%STBC_MIMO_OFDM.m
%这是一个基于空时分组编码的 MIMO_OFDM 通信系统的仿真设计。
%此系统包括 QPSK 调制解调、IFFT 调制、空时编解码、基于训练符号的信道估计等通信模块。


clear all
close all
clc

%+++++++++++++++++++++++++++变量++++++++++++++++++++++++++++++
i=sqrt(-1);
IFFT_bin_length=512;          %傅里叶变换抽样点数目
carrier_count=100;            %子载波数目
symbols_per_carrier=66;       %符号数/载波
cp_length=10;                 %循环前缀长度
addprefix_length=IFFT_bin_length+cp_length;
M_psk=4;
bits_per_symbol=log2(M_psk); %位数/符号
%[x1 x2;-x2* x1*] 二天线发送矩阵
% O=[1 2;-2+j 1+j];
```

```
    %[x1 -x2 -x3;x2* x1* 0;x3* 0 x1*;0 -x3* x2*] 三天线发送矩阵
    O=[1 -2 -3;2+j 1+j 0;3+j 0 1+j;0 -3+j 2+j];
    co_time=size(O,1);
    Nt=size(O,2);                %发射天线数目
    Nr=2;                        %接收天线数目
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %+++++++++++++++++++++++++++发射机++++++++++++++++++++++++++++++
    disp('--------------start-------------------');
    num_X=1;
    for cc_ro=1:co_time
        for cc_co=1:Nt
            num_X=max(num_X,abs(real(O(cc_ro,cc_co))));
        end
    end

    co_x=zeros(num_X,1);

    for con_ro=1:co_time
        for con_co=1:Nt     %用于确定矩阵“O”中元素的位置，符号以及共轭情况
            if abs(real(O(con_ro,con_co)))~=0
                delta(con_ro,abs(real(O(con_ro,con_co))))
=sign(real(O(con_ro,con_co)));
                epsilon(con_ro,abs(real(O(con_ro,con_co))))=con_co;
                co_x(abs(real(O(con_ro,con_co))),1)=co_x(abs(real
(O(con_ro,con_co))),1)+1;
                eta(abs(real(O(con_ro,con_co))),co_x(abs(real
(O(con_ro,con_co))),1))=con_ro;
                coj_mt(con_ro,abs(real(O(con_ro,con_co))))
=imag(O(con_ro,con_co));
            end
        end
    end

    eta=eta.';
    eta=sort(eta);
    eta=eta.';

    % 坐标: (1 to 100) + 14=(15:114)
    carriers = (1: carrier_count) + (floor(IFFT_bin_length/4) -
floor(carrier_count/2));
    % 坐标 : 256 - (15:114) + 1= 257 - (15:114) = (242:143)
    conjugate_carriers=IFFT_bin_length-carriers+2;
    tx_training_symbols=training_symbol(Nt,carrier_count);
    baseband_out_length = carrier_count * symbols_per_carrier;

    snr_min=3;                                    %最小信噪比
    snr_max=15;                                   %最大信噪比
    graph_inf_bit=zeros(snr_max-snr_min+1,2,Nr);      %绘图信息存储矩阵
    graph_inf_sym=zeros(snr_max-snr_min+1,2,Nr);
```

```
for SNR=snr_min:snr_max
  clc
  disp('Wait until SNR=');disp(snr_max);
  SNR
  n_err_sym=zeros(1,Nr);
  n_err_bit=zeros(1,Nr);
  Perr_sym=zeros(1,Nr);
  Perr_bit=zeros(1,Nr);
  re_met_sym_buf=zeros(carrier_count,symbols_per_carrier,Nr);
  re_met_bit=zeros(baseband_out_length,bits_per_symbol,Nr);

  %生成随机数用于仿真
  baseband_out=round(rand(baseband_out_length,bits_per_symbol));
  %二进制向十进制转换
  de_data=bi2de(baseband_out);
  %PSK 调制
  data_buf=pskmod(de_data,M_psk,0);
  carrier_matrix=reshape(data_buf,carrier_count,symbols_per_carrier);
  %取数为空时编码做准备，此处每次取每个子载波上连续的两个数
  for tt=1:Nt:symbols_per_carrier
    data=[];
    for ii=1:Nt
    tx_buf_buf=carrier_matrix(:,tt+ii-1);
    data=[data;tx_buf_buf];
    end

    XX=zeros(co_time*carrier_count,Nt);
    for con_r=1:co_time                              %进行空时编码
      for con_c=1:Nt
        if abs(real(O(con_r,con_c)))~=0
          if imag(O(con_r,con_c))==0
            XX((con_r-1)*carrier_count+1:con_r*carrier_count,con_c)
=data((abs(real(O(con_r,con_c)))-1)*carrier_count+1:abs(real(O(con_r,con_c)
))...
              *carrier_count,1)*sign(real(O(con_r,con_c)));
          else
            XX((con_r-1)*carrier_count+1:con_r*carrier_coun
t,con_c)=conj(data((abs(real(O(con_r,con_c)))-1)*carrier_count+1:abs(real(O
(con_r,con_c)))...
              *carrier_count,1))*sign(real(O(con_r,con_c)));
          end
        end
      end
    end                                              %空时编码结束

    XX=[tx_training_symbols;XX];                     %添加训练序列

    rx_buf=zeros(1,addprefix_length*(co_time+1),Nr);
    for rev=1:Nr
```

```
        for ii=1:Nt
          tx_buf=reshape(XX(:,ii),carrier_count,co_time+1);
          IFFT_tx_buf=zeros(IFFT_bin_length,co_time+1);
          IFFT_tx_buf(carriers,:)=tx_buf(1:carrier_count,:);
          IFFT_tx_buf(conjugate_carriers,:)=conj(tx_buf(1:carrier_
count,:));
          time_matrix=ifft(IFFT_tx_buf);
          time_matrix=[time_matrix((IFFT_bin_length-cp_length+1):
IFFT_bin_length,:);time_matrix];
          tx=time_matrix(:)';
    %++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %++++++++++++++++++++++++++信道++++++++++++++++++++++++++++++
          %d=randint(1,4,[1,7]);                          %4 多经信道模拟
          %a=randint(1,4,[2,7])/10;
          tx_tmp=tx;
          d=[4,5,6,2;4,5,6,2;4,5,6,2;4,5,6,2];
          a=[0.2,0.3,0.4,0.5;0.2,0.3,0.4,0.5;0.2,0.3,0.4,0.5;0.2,0.3,
0.4,0.5];
          for jj=1:size(d,2)
            copy=zeros(size(tx)) ;
            for kk = 1 + d(ii,jj): length(tx)
              copy(kk) = a(ii,jj)*tx(kk - d(ii,jj)) ;
            end
            tx_tmp=tx_tmp+copy;
          end

          txch=awgn(tx_tmp,SNR,'measured');              %添加高斯白噪声
          rx_buf(1,:,rev)=rx_buf(1,:,rev)+txch;
        end
    %++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %++++++++++++++++++++++++++接收机++++++++++++++++++++++++++++++
      rx_spectrum=reshape(rx_buf(1,:,rev),addprefix_length,co_time+1);
      rx_spectrum=rx_spectrum(cp_length+1:addprefix_length,:);
      FFT_tx_buf=zeros(IFFT_bin_length,co_time+1);
      FFT_tx_buf=fft(rx_spectrum);
      spectrum_matrix=FFT_tx_buf(carriers,:);
      Y_buf=(spectrum_matrix(:,2:co_time+1));
      Y_buf=conj(Y_buf');

      spectrum_matrix1=spectrum_matrix(:,1);
      Wk=exp((-2*pi/carrier_count)*i);
      L=10;

      p=zeros(L*Nt,1);
      for jj=1:Nt
          for l=0:L-1
              for kk=0:carrier_count-1
                  p(l+(jj-1)*L+1,1)=p(l+(jj-1)*L+1,1) +spectrum_matrix1
(kk+1,1) *conj(tx_training_symbols(kk+1,jj))*Wk^(-(kk*l));
```

```
            end
        end
    end
    h=p/carrier_count;

    H_buf=zeros(carrier_count,Nt);
    for ii=1:Nt
       for kk=0:carrier_count-1
          for l=0:L-1
             H_buf(kk+1,ii)=H_buf(kk+1,ii)+h(l+(ii-1)*L+1,1)*Wk^(kk*l);
          end
       end
    end
    H_buf=conj(H_buf');

    RRR=[];
    for kk=1:carrier_count
       Y=Y_buf(:,kk);
       H=H_buf(:,kk);
       for co_ii=1:num_X
          for co_tt=1:size(eta,2)
             if eta(co_ii,co_tt)~=0
                if coj_mt(eta(co_ii,co_tt),co_ii)==0
                   r_til(eta(co_ii,co_tt),:,co_ii)= Y(eta
(co_ii,co_tt),:);
                   a_til(eta(co_ii,co_tt),:,co_ii)=conj (H(epsilon
(eta(co_ii,co_tt),co_ii),:));
                else
                   r_til(eta(co_ii,co_tt),:,co_ii)=
conj(Y(eta(co_ii,co_tt),:));
                   a_til(eta(co_ii,co_tt),:,co_ii)
=H(epsilon(eta(co_ii,co_tt),co_ii),:);
                end
             end
          end
       end

       RR=zeros(num_X,1);

       for iii=1:num_X                          %接收数据的判决统计
          for ttt=1:size(eta,2)
             if eta(iii,ttt)~=0
                RR(iii,1)=RR(iii,1)+r_til(eta(iii,ttt),1,iii)
*a_til(eta(iii,ttt),1,iii)*delta(eta(iii,ttt),iii);
             end
          end
       end

       RRR=[RRR;conj(RR')];
    end
    r_sym=pskdemod(RRR,M_psk,0);
```

```
        re_met_sym_buf(:,tt:tt+Nt-1,rev)=r_sym;
      end
    end

    re_met_sym=zeros(baseband_out_length,1,Nr);

    for rev=1:Nr
      re_met_sym_buf_buf=re_met_sym_buf(:,:,rev);
      re_met_sym(:,1,rev)= re_met_sym_buf_buf(:);
      re_met_bit(:,:,rev)=de2bi(re_met_sym(:,1,rev));

      for con_dec_ro=1:baseband_out_length
        if re_met_sym(con_dec_ro,1,rev)~=de_data(con_dec_ro,1)
          n_err_sym(1,rev)=n_err_sym(1,rev)+1;
          for con_dec_co=1:bits_per_symbol
            if re_met_bit(con_dec_ro,con_dec_co,rev)~=
baseband_out(con_dec_ro,con_dec_co)
              n_err_bit(1,rev)=n_err_bit(1,rev)+1;
            end
          end
        end
      end
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    %+++++++++++++++++++++++++++误码率计算++++++++++++++++++++++++++++++
      graph_inf_sym(SNR-snr_min+1,1,rev)=SNR;
      graph_inf_bit(SNR-snr_min+1,1,rev)=SNR;
      Perr_sym(1,rev)=n_err_sym(1,rev)/(baseband_out_length);
      graph_inf_sym(SNR-snr_min+1,2,rev)=Perr_sym(1,rev);
      Perr_bit(1,rev)=n_err_bit(1,rev)/
(baseband_out_length*bits_per_symbol);
      graph_inf_bit(SNR-snr_min+1,2,rev)=Perr_bit(1,rev);
    %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    end
  end

  %+++++++++++++++++++++++++++性能仿真图++++++++++++++++++++++++++++++
  for rev=1:rev
   x_sym=graph_inf_sym(:,1,rev);
   y_sym=graph_inf_sym(:,2,rev);
   subplot(Nr,1,rev);
   semilogy(x_sym,y_sym,'b-*');
   axis([2 16 0.0001 1]);
   xlabel('信噪比/dB');
   ylabel('误码率');
   grid on
   %hold on
  end

  %hold off
```

```
%for rev=1:rev
 %x_bit=graph_inf_bit(:,1,rev);
 %y_bit=graph_inf_bit(:,2,rev);
 %subplot(2,1,2);
 %semilogy(x_bit,y_bit,'k-v');
 %axis([2 16 0.0001 1]);
 %xlabel('SNR, [dB]');
 %ylabel('Bit Error Probability');
 %grid on
 %hold on
%end
%hold off
disp('--------------end-------------------');
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
% ************************beginning of file*****************************
% compoversamp2.m
%
% 此函数实现"sample"倍升采样
%

function [iout,qout] = compoversamp2(iin, qin, sample)

%+++++++++++++++++++++++variables++++++++++++++++++++++++++++
% iin     输入序列实部
% qin     输入序列虚部
% iout    输出序列实部
% qout    输出序列虚部
% sample  升采样的倍数
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[h,v] = size(iin);

iout = zeros(h,v*sample);
qout = zeros(h,v*sample);

iout(:,1:sample:1+sample*(v-1)) = iin;
qout(:,1:sample:1+sample*(v-1)) = qin;

%************************end of file*********************************
```

14.7 本章小结

MIMO-OFDM 系统利用 MIMO 和 OFDM 两者技术具有的特点，将空间分集、时间分集以及频率分集有机的结合起来，从而能够大大提高无线通信系统的信道容量和传输速率，被业界认为是构建未来宽带无线通信系统最关键的物理层传输方案。本章首先对

MIMO 系统结构、空时编码技术做了阐述，然后围绕系统模型、系统性能、参数设置几个方面，介绍了基于 STBC 的 MIMO-OFDM 通信系统设计，并给出了完整的仿真程序代码，希望读者通过学习，可以很好地理解 MIMO-OFDM 系统特点和仿真原理，做到深入浅出、学以致用。