清 华 大 学

综 合 论 文 训 练

题目：协作通信中预编码算法设计和实现

系　　别：电子工程系

专　　业：电子信息科学与技术

姓　　名：赵涛

指导教师：牛志升　教授

2012 年 6 月 11 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　　期：_____

中文摘要

下一代的无线通信系统将充分利用多天线技术提高系统容量，预编码技术可以有效抑制小区间用户干扰，保证用户的服务质量。软件无线电平台因其低成本灵活易配置的特点，成为下一代无线通信系统设计实现与算法验证的良好平台。本文研究了若干预编码算法和信道估计算法，并介绍了我们在开源软件无线电平台 GNU Radio 和通用软件无线电外设 (USRP) 上设计实现的带有接收端信道估计、估计信息反馈、发送端协作预编码的多天线系统。我们在软件模拟信道和真实的无线环境中验证了算法，评估了系统性能。

关键词：多天线；预编码；信道估计；GNU Radio；通用软件无线电外设

ABSTRACT

In the next generation wireless communication system, MIMO technology will be exploited to increase the capacity. Coordinated precoding can be used to efficiently mitigate inter-cell interference and meet users' QoS requirements. Software Defined Radio(SDR) has become a good platform for designing, evaluating and implementing new wireless system because of its low cost, flexibility and configurability. We in this thesis investigate several precoding and channel estimation algorithms, and describe our MIMO system implementation equipped with channel estimation, channel information feedback and coordinated precoding on open source SDR platform GNU Radio and USRP. We verify our algorithms and evaluate our MIMO system in both simulated channel and real wireless environment.

Keywords: MIMO; precoding; channel estimation; GNU Radio; USRP

目 录

主要符号对照表

MIMO	多输入多输出 (Multiple Input Multiple Output)
CoMP	协作多点通信 (Coordinated Multi-Point Communication)
SNR	信噪比
SINR	信干噪比
SDR	软件无线电 (Software Defined Radio)
USRP	通用软件无线电外设 (Universal Software Radio Peripheral)
PSK	相移键控
RRC	根号升余弦
$\mathbf{H}$	信道矩阵
$(\cdot)^H$	矩阵的共轭转置
$(\cdot)^T$	矩阵的转置
$\|\|\cdot\|\|_F$	矩阵的 Frobenius 范数
$\mathrm{tr}\{\cdot\}$	矩阵的迹 (trace)

第 1 章　引言

1.1　课题研究背景、内容和意义

移动通信网络以其灵活方便的接入，已经在大众生活中普及。近年来智能手机等终端日益流行，网络应用尤其是多媒体应用不断涌现，人们对网络容量的需求持续增长。传统蜂窝网络的架构已经难以满足未来人们对高速率业务的需求。根据演进中的 4G 标准，下一代蜂窝网络将采用多天线技术（MIMO，多输入多输出），通过空间复用极大地提升系统容量。

在多天线系统中，预编码技术会有助于保证蜂窝网络中小区边缘用户的服务质量。传统的蜂窝网络中，小区边缘的用户距离小区基站较远，所接收到的信号较弱，同时由于距离相邻小区的基站较近，还会接收到较强的相邻基站发射的信号（干扰）。图 1.1显示了小区间干扰的典型场景。小区边缘用户往往处在干扰受限的条件下，服务质量难以得到保证。在下一代蜂窝网络中，相邻小区的基站可以进行协作，根据当前的信道状态通过在基站处对要发送的数据做协作预编码，抵消信道中的影响，可以使用户接收到干扰抑制后的信号，从而提升用户的服务质量。[1]

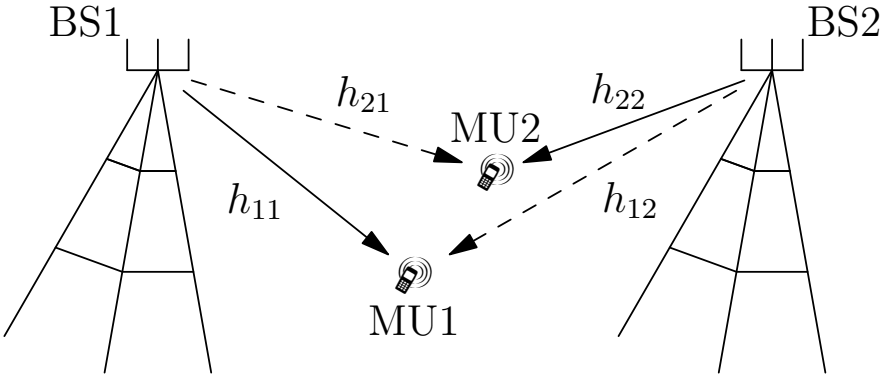

图 1.1　蜂窝网络中的小区间干扰场景示意图

采用预编码算法来抑制相邻小区干扰的多小区多天线系统，需要考虑信道估计、反馈信道、基站间同步、协作预编码等，相比传统蜂窝网络在系统结构上更为复杂。对这样一个新系统进行平台实现、算法验证、系统测试，是一个既有挑战性又有现实意义的问题。对此，软件无线电技术可以大有作为。近年

来逐渐受到重视的软件无线电 (SDR)技术，采用软件来实现各种信号处理算法、包结构的设计等，配合通用的硬件外设，可以用来快速的搭建真实无线环境中可用的通信系统，并且可以灵活方便地进行系统配置、算法测试。在现有的软件无线电平台中，GNU Radio[2] 因其开放源代码、使用通用编程语言开发等特点更便于快速开发实现出低成本的通信系统，受到学术界、工业界与业余爱好者的欢迎，得到了广泛的应用。

在这样的背景下，研究协作预编码算法，并在 GNU Radio 软件无线电平台上设计实现基于预编码的多天线系统，模拟多小区多天线系统中的相邻小区干扰抑制场景，具有很强的现实意义。

1.2 国内外研究现状

多天线技术是近年来无线通信领域研究的热点之一。文献 [3] 给出了多天线系统在瑞利衰落信道中的香农公式，指出了多天线技术可以带来很大的容量与频谱效率的提升。文献 [4] 指出了 MIMO-OFDM 系统在无线局域网 (WLAN)、4G 移动蜂窝网络中的应用前景和潜在挑战，特别指出了目前还缺少硬件实现上的研究的现状。

对于多天线系统中的预编码算法国内外已有许多理论研究。被广泛讨论的预编码算法有迫零 (ZF)算法、最小均方误差 (MMSE)算法等线性算法，以及基于脏纸编码 (DPC)的非线性算法等。[5,6] 这些算法需要大量的反馈信息，在实际中可能成为系统性能的瓶颈。一些文献基于有限反馈的限制条件，提出使用码本[7] 或改进了的旋转码本[8] 来实现预编码。此外还有一些迭代算法，文献中对其在多小区场景下是否可以分布式实现也作了考查。[9]

信道估计的准确性对预编码算法的性能有很大影响。目前理论研究中提出的多天线系统中的信道估计算法主要考虑基于导频（训练序列）的信道估计，具体算法有最小二乘 (LS)、缩放的最小二乘 (SLS)、最小均方误差 (MMSE)等。在基于导频的信道估计中，训练序列的选取对估计误差也有直接影响。文献中对最优训练序列所需要满足的条件也有研究。[10]

文献中可以看到软件无线电平台上的一些多天线系统原型实现。文献 [11] 使用 NI 公司的射频前端硬件和 LabView 软件环境实现了一个 MIMO-OFDM 系统原型。它采用基于导频的信道估计，但没有使用预编码，取而代之的是在接

收端对接收信号做迫零线性滤波。为了绕开同步问题，该系统中发射机和接收机的时钟线是直接相连的。在开源的软件无线电平台 GNU Radio 上，有文献实现了空时块编码 (STBC)和最大比合并 (MRC)的 MIMO 系统。[12] 另有文献使用 GNU Radio 和 USRP2 硬件外设实现了利用 Alamouti 编码的 2×2 多天线系统。[13] 我们实验室在自己的 GNU Radio 和 USRP 平台做了一些有关中继协作的研究。师兄王晓磊、徐洁、姜之源，师姐郭雪莹先后在该平台上实现了中继信道、中继合并算法和能效优先的中继选择算法。[14–16]

综上所述，现有研究给出了若干预编码和信道估计理论算法和仿真分析，但并没有实际系统的实现测试。现有软件无线电平台上已有多天线系统的初步实现，但没有涉及预编码算法。本实验室现有研究集中在基于中继的协作通信系统方面，还没有基于协作多点通信 (CoMP)、基站协作的系统设计。

1.3 毕设任务

我的毕业设计目标是在开源的软件无线电平台 GNU Radio 上实现一个带有预编码的多天线系统。具体内容可以分为算法调研分析和系统设计实现两个部分。

在算法调研分析部分，我会调研现有预编码算法、信道估计算法的原理和特点，分析其性能指标、适用范围等。这部分工作对平台上的算法实现有指导性意义。

在系统设计实现方面，我会熟悉 GNU Radio 上已有的代码，领会设计原则和步骤。在此基础上我将设计实现带有信道估计、实时反馈、协作预编码的多天线系统，模拟多天线多小区蜂窝网络基站协作进行干扰抑制的场景。我会利用软件模拟和硬件外设 USRP 在仿真信道和实际无线环境下进行算法验证和系统测试。

1.4 本文章节安排

本文的章节安排如下：

第 2 章包含理论算法的调研分析。我们对理论文献中提出的若干预编码算法和信道估计算法进行分析对比，指出各个算法的优缺点和适用场合。

第 3 章介绍我们使用的软件无线电平台 GNU Radio 以及硬件外设 USRP 的情况。硬件外设 USRP 的主要参数和特性会被给出，GNU Radio 的软件架构会被介绍。特别地我们会分析 GNU Radio 中已有的一个简单收发系统。

第 4 章给出带有信道估计、反馈信道、协作预编码的单发单收系统和多天线系统的设计实现。我们采用的信道估计算法、反馈方式、预编码算法、系统框架会得到详述。我们设计实现的同步算法、导频帧结构、仿真信道也会被给出。

第 5 章将介绍我们设计实现的单发单收系统和多天线系统在仿真信道以及真实无线环境下的测试结果，系统设置、测试结果以及结果分析会被给出。

第 6 章总结我们完成的工作，并指出进一步工作的方向。

第 2 章　理论与算法

本章从理论角度分别给出预编码算法和信道估计算法面对的问题的数学描述，列出具体算法，并对各个算法做出总结分析。

2.1　预编码算法

2.1.1　问题形成

我们考虑多小区多天线蜂窝网络的场景。记基站数目为 N_B，移动用户终端数目为 K，每个基站处的天线数为 N_t，每个用户终端处的天线数为 N_r。

考虑下行链路，第 k 个用户接收到的信号除了自身有用的信号外，还包括其他用户信号带来的干扰，以及环境噪声。数学上可以表示为：

$$\mathbf{y}_k = \mathbf{H}_k^{(b_k)}\mathbf{x}_k + \sum_{j\neq k}\mathbf{H}_k^{(b_j)}\mathbf{x}'_{jk} + \mathbf{z}_k. \tag{2-1}$$

其中 $\mathbf{H}_k^{(b_j)}$ 为 $N_r \times N_t$ 的矩阵，指的是第 j 个用户关联的基站 (b_j) 到第 k 个用户的信道。$\mathbf{x}'_{jk}$ 代表第 j 个用户的有用信号传播到第 k 个用户处的结果。由于小区边缘用户距离相邻基站距离相近，所以来自其他小区的干扰会比较严重，其服务质量相对较差。

由于在下行链路中基站处可以很好的同步，所以小区间干扰问题可以通过协作预编码来解决。使用预编码，我们有：

$$\mathbf{x}_k = \mathbf{T}_k\mathbf{d}_k \tag{2-2}$$

其中 $\mathbf{d}_k$ 为第 k 个用户所需的数据，可以是一个 $L\times 1$ 的向量，这里 L 表示每个用户一次传输的符号数。$\mathbf{T}_k$ 为预编码矩阵，可以用来对抗衰落信道的影响同时抑制相邻小区带来的干扰信号。如果所有用户的数据在各个基站间同步，$\mathbf{T}_k$ 为 $N_tN_B \times L$ 维矩阵，这被称为信号级的协作。有的算法不要求用户数据在各个基站间同步，这时 $\mathbf{T}_k$ 的维数可以降到 $N_t \times L$，更重要的是有可能得到分布式的实现，这种情形称为波束成形级的协作预编码。

我们记 $\mathbf{T}=[\mathbf{T}_1,\ldots,\mathbf{T}_K]$，$\mathbf{H}_k=[\mathbf{H}_k^{(1)},\ldots,\mathbf{H}_k^{(N_B)}]$，再把 K 个用户接收的信号写在一起 $\mathbf{y}=[\mathbf{y}_1,\ldots,\mathbf{y}_K]^T$，得到整个系统的矩阵描述：

$$\mathbf{y}=\mathbf{HTd}+\mathbf{z} \tag{2-3}$$

预编码算法的设计问题即为已知信道状况 $\mathbf{H}$ 的情况下设计合适的预编码矩阵 $\mathbf{T}$，使得接收端收到的信号 $\mathbf{y}$ 接近于所需的数据 $\mathbf{d}$。实际系统中 $\mathbf{H}$ 可以通过一些途径来获得，例如频分双工 (FDD)系统可以反馈信道由移动用户将测量出的下行信道信息反馈回基站，时分双工 (TDD)系统中可以借助上下行信道的互惠性由基站测量上行信道状况得到[9]。

2.1.2 迫零算法

迫零 (Zero Forcing, ZF)算法是一种简单的预编码算法。迫零算法给出的预编码矩阵就是 $\mathbf{H}$ 的伪逆乘上一个常数值：

$$\mathbf{T}=c_0\mathbf{H}^H(\mathbf{H}\mathbf{H}^H)^{-1}. \tag{2-4}$$

2.1.3 MMSE 算法

基于最小均方误差 (MMSE)准则，得到的预编码矩阵为：

$$\mathbf{T}=c_0\mathbf{H}^H(\mathbf{H}\mathbf{H}^H+\frac{\sigma^2}{P_T}\mathbf{I})^{-1} \tag{2-5}$$

2.1.4 结合脏纸编码的预编码算法

脏纸编码 (DPC)是信息论中的一种编码手段，使用它可以将已知会带来的干扰预先消除，使得一些用户对另一些用户“隐藏”。它是一种非线性的编码，使用脏纸编码可以使性能达到理论容量极限，而且不会有功率损失。

脏纸编码的实现复杂，实际中可用一些近似算法，如 THP(Tomlinson-Harashima Precoding)算法。

2.1.5 基于码本的预编码算法

基于码本的预编码需要收发双方事先约定好一组待选取的预编码矩阵称为“码本”：

$$\mathscr{T}=\{\mathbf{T}_0,\mathbf{T}_1,\ldots,\mathbf{T}_{2^B-1}\} \tag{2-6}$$

其中 B 为反馈比特数，$\mathbf{T}_i$ 均为酉矩阵，即满足 $\mathbf{T}_i^H\mathbf{T}_i = \mathbf{I}$。

接收端估计出信道值后，根据某个准则选出最优的预编码矩阵，将其下标（序号）反馈回发送端，这样只需在反馈信道传输 B 个比特，适合反馈信道容量有限的场景。

文献中还提出了一种旋转码本的方案，它需要预先准备多个码本，但每次使用其中的一个和一个历史上最优的预编码矩阵。经过码本的旋转轮换，我们可以在不增加反馈比特数的前提下使用一个更大的虚拟码本，从而提高预编码性能。

2.1.6 迭代算法

针对多小区中下行链路的波束成行问题，还有一些迭代算法，如波束成形-功率迭代算法以及基于迭代函数求值的算法。算法的步骤参见附录 A。

2.1.7 算法对比分析

基于脏纸编码的非线性算法可以达到理论极限，但实现复杂。线性算法可以用较低的复杂度取得足够好的性能，其中迫零算法在高 SNR 情况下和 MMSE 算法性能接近，但在低 SNR 情况下性能有明显差距。以上算法都基于发送端有完美的信道信息的前提，在实际系统中信道信息估计的误差将影响算法的效果。

在反馈信道容量受限的条件下，反馈完全的信道信息可能变得不再实际，而基于码本的预编码则可以提供较好的性能。性能好坏与反馈比特数多少直接相关。在一定的反馈比特数下，可以采取旋转码本的机制来改善性能，但这增加了策略上的复杂度。

多小区中迭代算法可以实现波束成行级的协作预编码，算法实现的复杂度和算法是否可以分布式实现是衡量算法的重要指标。在实际中，往往要在迭代次数和优化前提设置之间进行折衷。

2.2 信道估计算法

前面已经提及，预编码算法是建立在信道状况 $\mathbf{H}$ 的值已知的前提下的。无论是接收端测量信道将结果反馈回发送端，还是发送端测量信道再利用信道互惠性，都需要做信道估计。信道估计足够准确才能保证预编码的有效性。本节将给出多天线系统中信道估计的数学描述和若干算法。

2.2.1 问题形成

信道估计可以分为两大类，一类基于训练序列：发送端发射先验确定的训练序列[①]，接收端根据训练序列和接收到的实际信号计算信道的估计值；另一类称为“盲估计”，它不使用训练序列，直接从接收信号的信息中进行估计。多天线系统中一般考虑前一类方法。

记训练序列 $\mathbf{P}=[\mathbf{p}_1,\ldots,\mathbf{p}_N]$，为 $N_tN_B\times N$ 维矩阵，N 为训练序列的长度满足 $(N\geq N_tN_B)$。信道 $\mathbf{H}$ 为 $KN_r\times N_tN_B$ 维，理论分析中一般设定为瑞利 (Rayleigh)块衰落信道。接收端噪声用 $\mathbf{V}$ 来表示。这样各个接收天线得到的信号为

$$\mathbf{S}=\mathbf{HP}+\mathbf{V}. \tag{2-7}$$

信道估计问题即为在已知训练序列 $\mathbf{P}$ 以及接收信号 $\mathbf{S}$ 的条件下求信道 $\mathbf{H}$ 的估计值。容易得知对于单小区的场景，问题的描述也是类似的。下面为了简便，记 $t=N_tN_B, r=KN_r$ 。

2.2.2 最小二乘算法

根据最小二乘 (LS)准则得到的信道估计值为

$$\hat{\mathbf{H}}_{\mathrm{LS}}=\mathbf{SP}^{+}, \tag{2-8}$$

其中 $\mathbf{P}^{+}$ 为 $\mathbf{P}$ 的伪逆：

$$\mathbf{P}^{+}=\mathbf{P}^{H}(\mathbf{PP}^{H})^{-1}. \tag{2-9}$$

寻找最优的训练序列的问题可以归结为一个最优化问题：

$$\min_{\mathbf{P}} E\{||\mathbf{H}-\hat{\mathbf{H}}_{\mathrm{LS}}||_F^2\} \quad \text{subject to} \quad ||P||_F^2=\mathscr{P}. \tag{2-10}$$

我们的目标是最小化信道估计误差，前提条件是训练序列的功率限制（给定常数 $\mathscr{P}$）。求解这个最小化问题，可知最优训练序列所需的限制条件为

$$\mathbf{PP}^{H}=\frac{\mathscr{P}}{t}\mathbf{I} \tag{2-11}$$

也就是说，只要 $\mathbf{P}$ 每行都正交，且范数都为 $\sqrt{\mathscr{P}/t}$，这样的 P 都是最优的。

① 也称导频，在多天线系统中实为矩阵。本文中对这几个术语不做区分。

2.2.3 缩放的最小二乘算法

注意到最小二乘算法是最小方差的无偏估计，它并不能保证最小化均方误差 (MSE)。文献 [10] 提出了一种缩放的最小二乘算法，通过在最小二乘的估计值乘上一个缩放因子 γ 以进一步缩小估计误差。

$$\min_{\gamma} E\left\{||\mathbf{H}-\gamma\hat{\mathbf{H}}_{\mathrm{LS}}||_F^2\right\} \tag{2-12}$$

问题的最优解为

$$\gamma_o = \frac{\mathrm{tr}\{\mathbf{R_H}\}}{\sigma_n^2 r\,\mathrm{tr}\{(\mathbf{P}\mathbf{P}^H)^{-1}\}+\mathrm{tr}\{\mathbf{R_H}\}} \tag{2-13}$$

式中 $\mathbf{R_H}=E\{\mathbf{H}^H\mathbf{H}\}$ 为信道的自相关。

所以缩放最小二乘算法给出的信道估计值为

$$\hat{\mathbf{H}}_{\mathrm{SLS}} = \frac{\mathrm{tr}\{\mathbf{R_H}\}}{\sigma_n^2 r\,\mathrm{tr}\{(\mathbf{P}\mathbf{P}^H)^{-1}\}+\mathrm{tr}\{\mathbf{R_H}\}}\mathbf{S}\mathbf{P}^H(\mathbf{P}\mathbf{P}^H)^{-1}. \tag{2-14}$$

缩放的最小二乘算法下的最优训练序列和最小二乘算法下的相同。注意到缩放的最小二乘算法中需要 $\mathrm{tr}\{\mathbf{R_H}\}$，实际应用中可以用 $\mathrm{tr}\{\hat{\mathbf{H}}_{\mathrm{LS}}^H\hat{\mathbf{H}}_{\mathrm{LS}}\}$ 来代替。

2.2.4 最小均方误差算法

最小均方误差准则下的信道估计值为

$$\hat{\mathbf{H}}_{\mathrm{MMSE}} = \mathbf{S}(\mathbf{P}^H\mathbf{R_H}\mathbf{P}+\sigma_n^2 r\mathbf{I})^{-1}\mathbf{P}^H\mathbf{R_H}. \tag{2-15}$$

最优训练矩阵 $\mathbf{P}$ 应满足限制条件 ①

$$\mathbf{P}\mathbf{P}^H = \frac{1}{t}(\mathscr{P}+\sigma_n^2 r\,\mathrm{tr}\{\mathbf{R_H}^{-1}\})\mathbf{I}-\sigma_n^2 r\mathbf{R_H}^{-1}. \tag{2-16}$$

注意到当信道不相关或者 SNR 很高时，这个限制条件就退化为 (2-11)式。

2.2.5 算法对比分析

算法在估计准确性和需要的先验知识的多少之间存在折衷。最小二乘算法不需要信道和环境噪声的信息，在高 SNR 情况下可以提供比较好的信道估计，但在 SNR 较低时误差会较大。缩放的最小二乘算法在高 SNR 时的估计误差和

① 这个限制条件只在足够高的 SNR 条件下成立，在任意 SNR 条件下的限制条件参见[10]。

最小二乘算法的基本相同，在低 SNR 时的估计误差小于最小二乘算法的情形，但是这个算法需要 $\mathrm{tr}\{\mathbf{R_H}\}$ 的信息（或其估计值）。最小均方误差算法的估计性能总是优于最小二乘和缩放最小二乘，但其实现复杂，需要事先已知 $\mathbf{R_H}$（实际中需要进行松弛），而且在高 SNR 条件下性能提升有限。

第 3 章　GNU Radio 软件无线电平台介绍

本章介绍我们算法设计实现所用到的软件无线电平台 GNU Radio 以及相应的硬件外设 USRP。

3.1　USRP

USRP 意为通用软件无线电外设 (Universal Software Radio Peripheral)，它是 Ettus Research 公司[17] 的产品。它带有射频收发模块，可以通过 USB2.0 接口和 PC 机相连，从而可以将 PC 机上软件无线电基带信号转换为射频信号，也可以将接收到的射频信号转换为基带信号交给软件作进一步处理。图 3.1是 USRP 的一张实物照片，图 3.2显示了 USRP 典型的连接关系。使用 USRP，软件无线电系统可以应用在真实的无线环境中。软件无线电的系统架构使得硬件外设可以通用，我们可以在不改动硬件的条件下，通过软件编程实现不同的功能和算法，得到不同的应用。

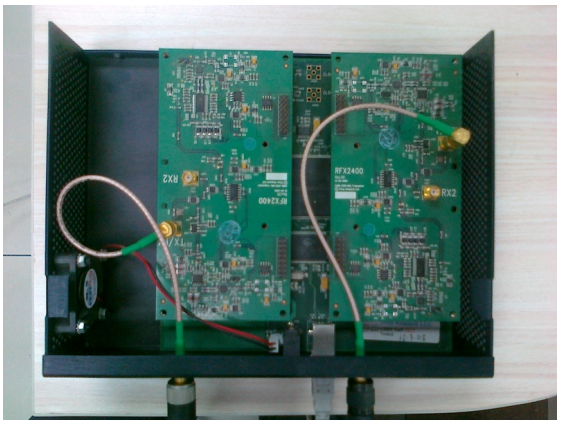

图 3.1　USRP 实物照片

如图 3.2所示，USRP 由母板和子板两部分构成。母板上搭载了一个 Altera Cyclone FPGA 芯片，用来做高速的数学运算。标准的 FPGA 映像中实现了两个数字下变频器 (DDC)和两个上变频器 (DUC)，完成信号在基带与中频之间的转换。DDC 会对中频信号进行抽取，得到适合传输到 PC 机的速率。DUC 其实是位于 AD9862 CODEC 芯片中，FPGA 只负责进行插值。FPGA 接到 USB 接口芯

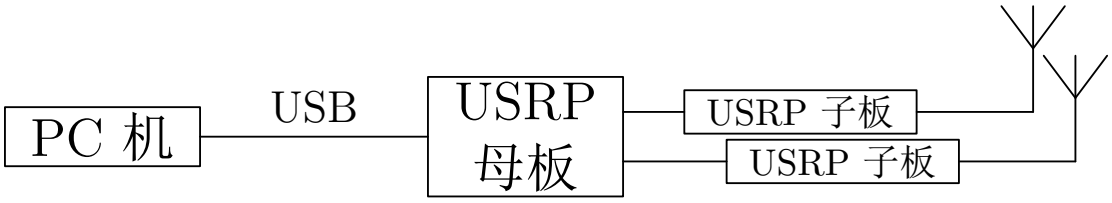

图 3.2　USRP 组成和连接

片 Cypress FX2 上，通过它连接到电脑上。注意基带采样率只能取一些离散的值，以使 USRP 有合适的插值率和抽取率，表 3.1 列出了所有有效的组合。

USRP 母板上搭载了一个 64 MHz 晶振作为内部时钟源。它的频率稳定度为20 PPM，在 2.4 GHz 频偏最大可达几十 kHz。注意设备工作一段时间后元件温度升高也会影响频率稳定度。要获得更稳定的时钟信号，需要修改硬件换用外部时钟源。

表 3.1　USRP 有效的插值率、抽取率与对应的基带采样率

插值率	抽取率	采样率 (Sample/s)
16	8	8 M
32	16	4 M
64	32	2 M
128	64	1 M
256	128	500 k
512	256	250 k

母板上带有四个数模转换器 (DAC)，参数均为每采样 14 bit，采样速率 128 MSample/s。此外还有四个模数转换器 (ADC)，参数均为每采样 12 bit，采样速率 64 MSample/s。通过 DAC，数据从母板发送到子板进而发射出去。ADC 则反过来，将子板收到的数据传输到母板。

USRP 母板上可以接入多种子板，覆盖的频率范围从 0 Hz 到 6 GHz。子板作为射频前端，负责模拟信号的发射和接收。我们实验室的 USRP 搭载的子板型号是 RFX2400，它的工作频率范围是 2.3–2.9 GHz。每个 RFX2400 子板有两个天线端口，一个是 RX2 端口只能作为接收端使用，另一个为 TX/RX 端口既可以用来发射又可以用来接收。值得注意的是 TX/RX 端口上接了一个声表面波 (SAW)滤波器，它在 2.4–2.483 GHz 频带内增益最大，所以使用 TX/RX 收发这个频段内的信号时效果最佳。每个 USRP 母板上可以接两个 RFX2400 子板，如

图 3.1所示。

USRP 可以工作在全双工模式下，拥有独立的发送和接收功能。唯一限制在于合起来在总线上的数据速率不大于32 MByte/s。USRP 在 USB2.0 接口上传输数据总是使用 16 位 I/Q 两路采样，因此这意味着采样速率需要不大于8 MSample/s。除了母板和子板外，USRP 设备提供一个 6V 4A 的电源适配器，将市电转换为 USRP 所需的电源。另外还提供一根 USB 连接线，用来接入 USB2.0 接口的 PC 机。USRP 不支持 USB1.1 接口。

3.2 UHD

UHD 意为通用软件无线电外设硬件驱动 (USRP Hardware Driver)，提供了 Ettus 公司的一些通用软件无线电外设（如 USRP、USRP2 等）的设备驱动。UHD 是开放源代码的，除了驱动代码和固件代码外还包含了 USRP FPGA 设计的代码。这意味着 USRP 属于开源硬件，可以方便地修改定制硬件功能，满足特定场合的需求。UHD 编译安装后就可以使得通过 USB 数据线连接到 PC 机上的 USRP 被操作系统识别并正确使用。此外 UHD 提供了诸如 `uhd_find_devices` 和 `uhd_usrp_probe` 之类的命令行工具用来发现设备和探测硬件信息。

3.3 GNU Radio

3.3.1 软件架构

GNU Radio 是我们的软件无线电平台的软件开发环境，它是开源/自由软件，是自由软件基金会 GNU 项目的一部分。GNU Radio 提供了许多信号处理模块实现信号处理算法，更重要的提供了接口以开发更多的信号处理模块。GNU Radio 提供了将各个模块连接起来成为一个完整系统（术语称为“流图”）的方法，并实现了一个调度器 (scheduler)，使得数据可以在流图中的各个模块间流动，从而完成软件无线电的任务——信号的基带处理。

GNU Radio 的原生平台是 Linux 操作系统，其源代码仓库使用 Git 版本控制工具进行维护，代码构建工具使用的是 CMake[①]。我们在实验室的工作电脑

① 早期版本使用的构建工具是 GNU autotools，从 3.5 版本起逐步切换到 CMake 上。

（一台 IBM T41 笔记本、一台 Lenovo T60 笔记本和一个 Dell 台式机）上安装了 Linux 操作系统（Ubuntu 11.10 和 Fedora 16），使用 `build-gnuradio` 脚本[18] 编译安装了 GNU Radio 的 3.5.x 分支版本。我们跟进了一段时间代码库的改进和修复后，最终稳定在 3.5.3.2 版本上。

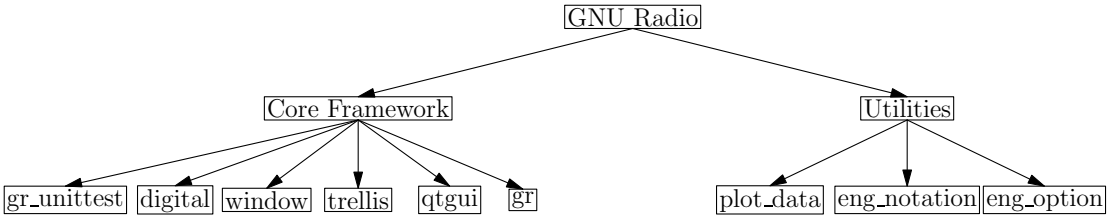

图 3.3　GNU Radio 核心框架和实用程序

如图 3.3 所示，GNU Radio 提供了包含数字调制解调、滤波器设计、音频视频处理算法等的核心框架，还实现了绘图、工程常用计数法选项等实用程序，为软件无线电开发者提供了一个容易上手、快速开发的平台。我们在实现基于预编码算法的多天线系统时，主要用到了 `gr`, `digital`, `eng_option`, `gr_unittest` 等组件，此外还实现了自己的组件 `niulab`。在 GNU Radio 上开发新模块时，既可以直接在 GNU Radio 代码树中添加，也可以新建独立的项目目录。两种方式各有优劣，前一种方式便于整合现有代码，可以方便地和上游互动，但每次编译安装都需要编译安装整个 GNU Radio 项目，后一种方式与之相对。我们在具体实现中使用到了这两种方式，对于后者还用到了 `gr-modtool` 工具[19] 简化项目建立、添加模块的操作。

前面已经提到了 GNU Radio 常用的两个名词“流图”(flowgraph)和“模块”(block)，二者的关系如下。流图中的节点称为模块，完成某一个特定的信号处理任务；若干个模块连接起来构成一个流图，可以完成系统任务如信号的调制、从 USRP 接收数据记录到文件中等。一个（有意义的）系统或应用由一个或更多的流图构成，一个流图由一个或多个模块组成。注意 GNU Radio 中流图不能出现环路，需要明确的源端和末端。

在 GNU Radio 上开发需要使用 C++ 和 Python 两门编程语言。C++ 被用来实现一些对性能要求较高的算法和模块。为了进一步提升性能，GNU Radio 一些模块实现中使用了 VOLK (Vector-Optimized Library of Kernels) 和 ORC(Oil Runtime Compiler) 库，利用 CPU 架构中的单指令多数据 (SIMD)指令集做向量化运算。使用 C++ 构造的模块都继承自 `gr_block` 或其子类，GNU Radio 提供

了 gr_sync_block、gr_sync_decimator 等模块，它们继承自 gr_block，并且定义好了输入输出端口的速率变换关系，方便开发者创建新的模块。使用 C++ 创建的模块，都会用到 Boost 库的智能指针 shared_ptr。它提供了对程序员透明的引用计数，会自动释放不再被使用的对象的内存空间，简化了内存管理的任务，尤其适合 C++ 和 Python 混合编程的架构。C++ 模块类的构造函数总是设成私有成员 (private)，通过友元函数 (friend)来构造类的实例。

Python 是一门通用的解释执行的面向对象的高级语言，在 GNU Radio 中用来连接模块，搭建流图或者层级模块 (hierachical block)。① 借助于 SWIG (Simplified Wrapper and Interface Generator, 简单包装程序与接口生成器)，Python 中可以调用 C++ 实现的模块。Python 语言内置了一些高级的数据结构（如字符串、列表、元组等），而且有功能丰富的标准库和第三方库（如文件操作、命令行参数处理等），使用它我们可以用较少的代码量实现完整的系统，加速开发效率；而且解释型的特点使得编码测试周期缩短，便于迭代开发。

SWIG 是一个用于生成 C/C++ 代码到脚本语言（如 Perl,Python,Tcl 等）的接口的实用工具，被 GNU Radio 项目用来简化 C++ 到 Python 的接口的生成。SWIG 可以自动处理好 C++ 中的字符串类、STL 中的向量等到 Python 中数据结构的映射。例如借助于 SWIG，C++ 中实现的 gr_vector_sink_c 模块可以在 Python 中用如下方式使用：

```
from gnuradio import gr
chn = gr.vector_sink_c()
```

GNU Radio 中模块接口常用的数据类型有复数、浮点数、整数和字符型等。复数类型用来表示复基带数据，在 C++ 中被定义为 gr_complex，它本质是 std::complex<float>的别名，在 Python 中就对应内置的复数类。字符型 (unsigned char 或 char) 常用来表示二进制序列，也用作标志输出。

GNU Radio 中还带有一个简单易用的 GRC(GNU Radio Companion) 组件，提供了类似 MATLAB Simulink 的图形化开发界面。通过 GRC 我们可以用鼠标拖拽点击选取模块搭建流图，它可以自动生成 Python 代码以运行流图。GNU Radio 模块需要提供 XML 文件描述以在 GRC 中使用，目前还有一些模块无法在 GRC 中使用，所以我们没有使用 GRC 开发带有预编码的多天线系统。

① 注意此处的模块与 Python 语言中的模块 (module)不是一个概念。Python 将一个脚本文件视为一个模块，其中可以定义多个类对应多个 GNU Radio 模块。

3.3.2 简单收发系统

在 GNU Radio 的 digital 组件中实现了一个简单的收发系统，支持以数据包为单位发送和接收数据。它利用了通用的调制模块和解调模块，可以支持高斯最小频率键控 (GMSK)、相移键控 (PSK，包括差分非差分的 BPSK、QPSK 等)、正交幅度调制 (QAM) 等多种调制方式。我们通过分析这个系统来深入地介绍 GNU Radio 上的系统设计，进而引出我们的算法设计和系统实现。

这个简单的收发系统利用了 C++ 和 Python 混合编程的软件架构。低层模块实现采用 C++ 语言，例如 `gr_chunks_to_symbols_bc` 将二进制数据映射到复数星座点上，`digital_constellation_receiver_cb` 将接收机接收到的星座点上的数据通过相位恢复和判决得到二进制数据。较高层的模块和系统实现都使用 Python 语言。通用的调制模块与解调模块在文件 gr-digital/python/generic_mod_demod.py 中实现。以数据包为单位的调制解调在文件 gr-digital/python/pkt.py 中实现。在 gr-digital/examples/narrowband/ 目录中，uhd_interface.py 对 USRP 源点模块和汇点模块做了包装，transmit_path.py 和 receive_path.py 分别实现了发送通道和接收通道（不包含发送到的目标模块和接收自的源模块），benchmark_tx.py 和 benchmark_rx.py 分别创建发射机和接收机的流图，其中发射机生成待发送的数据内容，接收机为接收到数据包的事件定义回调函数以实时输出接收情况。

完整的发射机结构如图 3.4 所示，其中每个框都对应一个 GNU Radio 模块，模块之间连线中点线代表字符型数据，实线代表复数型数据。用户数据先被打包送到消息源的消息队列中，数据包被通用的调制模块转换到星座点上，成为复基带数据。数据会被限幅，然后送到 USRP 发射出去或者写入文件。通用的调制模块会将数据字节拆分为 M 进制序列（对应 C++ 中的 `unsigned char` 或`char`类型，但只有低 $K = \log_2 M$ 位有效），经过可选的 Gray 映射和差分模块，将序列映射到星座点上，送给 RRC 滤波器做脉冲成型。

接收机的功能模块与发射机相对，但由于收发不同步，接收机还需要进行一系列的同步操作。接收机具体的结构框图如图 3.5。接收机的数据来源可以是 USRP 也可以是一个文件，从源端读取的数据会先进行滤波滤出关心的频带，然后将数据送到解调模块。解调后得到了二进制序列，接下来用一个相关器找到包头，将识别到的包存入最后的帧接收器。通用的解调模块完成同步和解调的功能：首先将滤波后的信号通过自动增益控制 (AGC)模块，然后恢复载波频

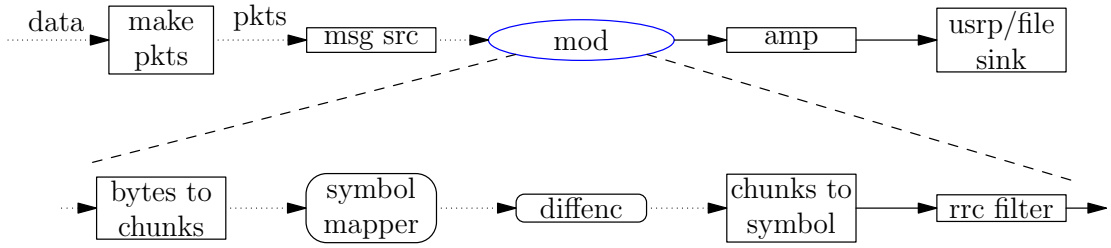

图 3.4　GNU Radio 实现的发射机结构框图

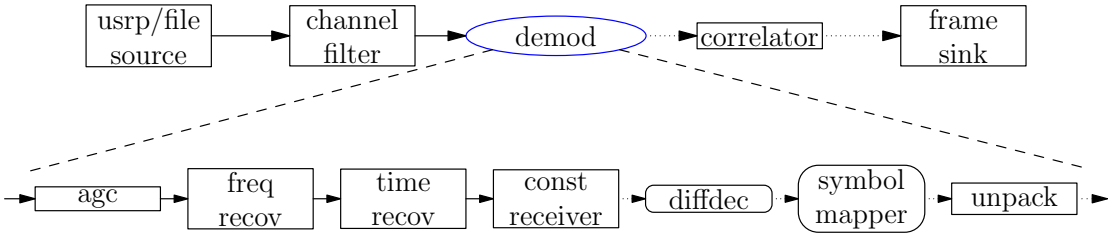

图 3.5　GNU Radio 实现的接收机结构框图

率，恢复位定时，之后做相位恢复并判决得到 M 进制序列，接下来做可选的差分译码和逆 Gray 映射，最后拆分得到二进制序列。

这个收发系统采用的数据包帧结构如图 3.6所示。帧头包括16 bit的序言(preamble)，64 bit用于识别用户的接入码，和两个相同的 16 bit域指示包长。接收端只有看到两个16 bit 完全相同时才接收这个包。这里的包长指的是载荷 (payload)和载荷的 CRC 的总长。CRC 总是32 bit长，默认情况下载荷长度是1500 Byte。载荷起始处的16 bit被用来指定包序号。接下来是帧结束符`'\x55'`，再后面还会有零到多个`'\x55'`作为填充，以使整个包调制后在 USB 接口上传输时恰好为512 Byte（USB 传输管道中包长的典型值）的整倍数。对于每符号采样数 (SPS)为 2 的 BPSK 调制，默认载荷长度下填充字节数为 1。①

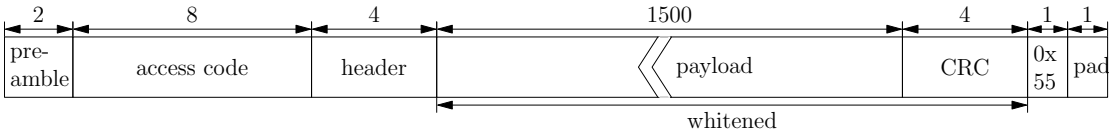

图 3.6　GNU Radio 实现的帧结构

① 具体的计算过程是：因为 USB 接口上每个采样是16 bit I/Q 两路共 4 个字节，所以512 Byte即对应 128 次采样。对于 BPSK，每符号比特数为 1，若每符号采样数为 2，则调制前一个包比特数应该为 64 的整数倍，包字节数应为 8 的整数倍。填充之前已有 $2+8+4+1500+4+1=1519$ 个字节，而 $1519 \mod 8 = 7$，所以只需填充一个字节。GNU Radio 中在packet_utils.py 中定义了函数`_npadding_bytes`实现了这个计算。

收发系统接收命令行 `--log` 选项，可以使用它将调制解调中的各个模块的输出写到文件中。借助于 GNU Radio 提供的 M 脚本，文件中的数据可以在 MATLAB 或者 Octave 中读取，可供进一步分析。这是系统设计实现中一个很好的分析与调试方式。

第 4 章　GNU Radio 上的设计实现

我们在 GNU Radio 中已经实现的简单收发系统的基础上，添加信道估计算法和预编码算法模块，实现了带有预编码的单发单收系统。为了实现信道估计，我们设计了导频信号使用的的帧结构。有了单发单收系统后，我们将信道估计和预编码扩充到多天线场景中，实现了带有预编码的多天线系统。本章将具体介绍我们系统的设计实现。

4.1　导频的帧结构

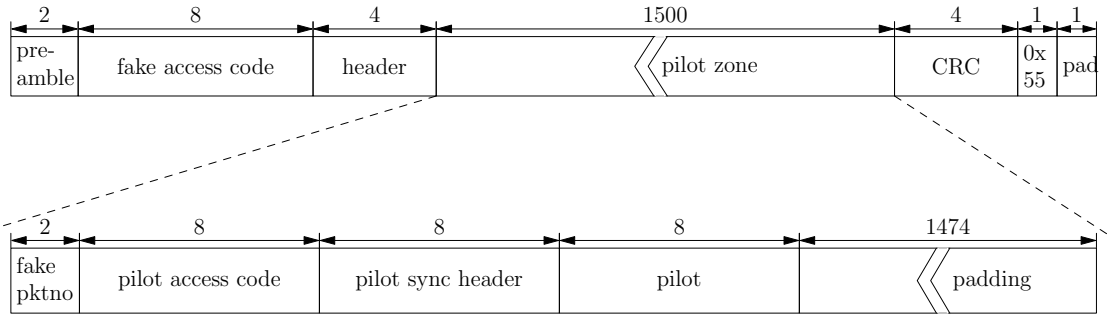

图 4.1　导频的帧结构

为了兼容已有的帧结构，重用现有代码，导频和用户数据的数据包有着类似的帧结构。导频的帧结构如图 4.1所示。首先是“伪造”的帧头，包括序言、伪造的接入码和四个字节的包长域。我们不使用用户真实的接入码以避免训练包被当作普通用户数据包解调接收。紧接着的两个字节是伪造的包序号，没有实际意义。接下来的24 Byte是真正有用的信号。8 Byte的导频接入码用于在发送端调制后模块将随后的导频同步头和导频信号转换为复数域的 0 和 1。8 Byte的导频同步头用于接收端信道估计模块寻找导频。后面的8 Byte的导频信号是真正的训练序列，用于接收端计算信道估计值。之后为填充字符`'\x55'`，使得导频与用户数据包载荷有同样的长度。接下来是 CRC 校验值、结束符和填充符，这样导频整个包长和数据包长相同，同为 512 字节的整倍数。与用户数据包相比较，我们将导频信号放在了对应于载荷的位置中。注意用户数据包的载荷及 CRC 会被白化 (whitening)，但导频信号的载荷部分出于训练的需要不能被白化。

我们采用了单极性的二进制序列做导频同步头和导频信号，以避免双极性序列到接收端时由于相位模糊可能带来的误判。在实际中接收端信道估计时，我们需要用一个幅度门限来判定星座点是 0 还是 1，我们将它作为模块的一个可调参数使用户可以根据实际信道情况对其微调。

我们没有采用在用户数据包里携带导频的方法。原因是我们采用发送端预编码的做法，接收端并不做操作抵消信道影响。由于更新了的信道信息反馈回发送端会有一定延时，所以导频包中的数据容易丢失。

4.2 单发单收系统

4.2.1 信道估计

在使用 C++ 实现信道估计时，我们采用了层次化结构，将模块与内部的信道估计算法分为两层。这样分层之后，使得修改算法与修改模块实现相分离，修改内部算法时可以不改动对外提供模块的代码。①

如图 4.2所示，我们的单发单收系统中实现了两个信道估计模块 `digital_mpsk_chn_est_cc` 和 `digital_probe_mpsk_chn_est_c`。它们都继承自 `gr_sync_block` 类，需要的参数都是三个：信道估计算法类型、导频同步头和导频序列。模块的输入信号都是两路，第一路用于对信道幅度进行估计，第二路对相位进行估计。两个模块的不同点在于模块输出：`digital_probe_mpsk_chn_est_c` 模块没有输出端口，适合用作流图末端检测模块，信道估计值可通过调用其成员函数 `chn`得到；`digital_mpsk_chn_est_cc` 则适于作为流图中间模块，它会输出信道估计值更新标志和当下最新的信道估计值送给下级模块。

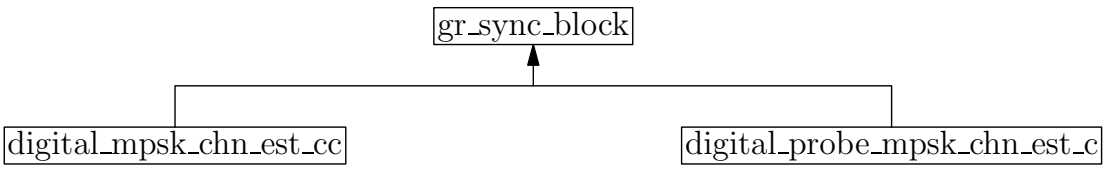

图 4.2 单发单收系统信道估计模块继承关系

① 另一方面，这种分层可能带来的劣势是向外暴露一个设置或者获取参数的功能时需要写多层代码。实际中需要处理好折衷关系。

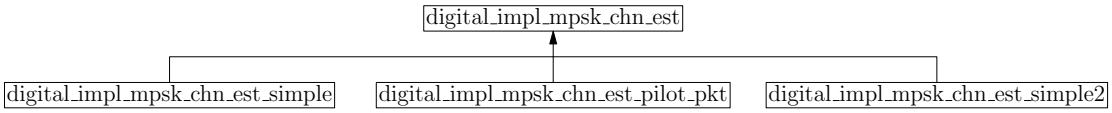

图 4.3　单发单收系统信道估计算法实现类继承关系

在算法的内部实现上，我们采用类的继承和派生支持多个信道估计算法。如图 4.3 所示，我们实现了 `digital_impl_mpsk_chn_est` 作为各个算法的基类。上层模块类中包含一个指向该基类的指针，以调用信道估计算法。在实现了的派生类中，`digital_impl_mpsk_chn_est_simple` 和 `digital_impl_mpsk_chn_est_simple2` 是两个概念性的简单实现，分别计算一定数目输入样点的滑动平均值和算术平均值作为信道估计值，它们没有利用导频信息，所以并不实用。`digital_impl_mpsk_chn_est_pilot_pkt` 是我们基于导频的信道估计使用最小二乘算法的实现。

我们算法的核心 `update` 函数实现了一个状态机：初始时处于同步阶段 (`STATE_SEARCH`)，我们检查最近的输入数据是否与导频的同步头匹配，若匹配则进入计算阶段 (`STATE_CALC`)。在计算阶段，我们会根据当前接收星座点更新信道估计值。在单发单收系统中，信道退化为标量 h。我们使用二进制导频序列，于是信道估计值就等于对导频为 1 时对应的星座点求算术平均。因此实现中我们利用变量`y1`记录星座点值的累加和，利用`d_calc_eff_cnt` 对累加次数计数。注意这里的星座点值是幅度输入和相位输入两路组合得到的复数值。当导频序列被全部考虑后，我们计算出平均值更新信道估计值寄存器`d_y1`，并且回到同步阶段，开始寻找新的同步头。我们的标志输出在同步阶段输出与导频同步头相比的错误比特数的相反数，在计算阶段输出已经考虑过的导频符号数。因此标志输出出现极大值对应着信道估计值做了一次更新。图 4.4给出了状态转换的示意图。

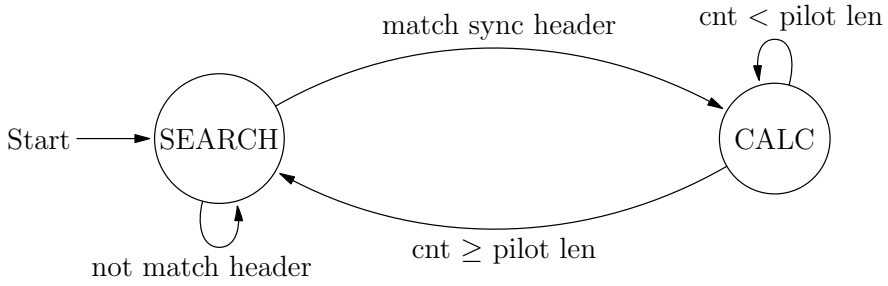

图 4.4　信道估计模块状态转换图

4.2.2 反馈信道

我们目前使用 UDP 数据包将接收端信道估计值反馈回发送端。GNU Radio 中实现了`gr_udp_sink`模块，可以用来向给定主机的给定端口发送 UDP 数据包，套接字的源端口是自动生成的临时端口号，我们只需指定数据单位大小、目标主机名或 IP 地址和目标端口即可。如下一行代码[①] 就在 Python 中创建了一个`gr_udp_sink`：

```
self.chn_udp_rx = gr.udp_sink(gr.sizeof_gr_complex,
   host, port)
```

将其输入连接到信道估计模块的输出，即可将信道估计值发送到指定的 UDP 端口。

在发送端使用`gr_udp_source`模块从 UDP 端口取出数据，即可得到反馈的信道估计值。模块的参数与`gr_udp_sink` 的相同，如下 Python 代码就创建了一个`gr_udp_source`：

```
self.fb_chn = gr.udp_source(gr.sizeof_gr_complex,
   host, port)
```

4.2.3 预编码

我们设计实现了`niulab_precoding_cc`模块在单发单收系统中做预编码。因为它的输入输出数据速率满足 1 : 1 关系，所以很自然地将它定义为`gr_sync_block`的派生类。它的输入输出端口数可以为 1 或者 2：必需的输入是数据输入，必需的输出是数据输出；此外信道估计值为可选输入，如果没有就假定为理想信道 $h = 1$，可选输出为标志输出，用于指示状态和数据分析。

在单发单收的情况下，信道矩阵 $\mathbf{H}$ 退化为一个标量 h，采用迫零算法，预编码矩阵退化为信道估计值的倒数 $1/\hat{h}$。也就是说预编码操作简化为输入数据与信道估计值两复数相除。不过我们实现中让预编码模块同时负责将调制了的导频同步头和导频序列转换为星座点上的 0/1 序列。为此它会以导频接入码和导频总长度（包括导频接入码、导频同步头和导频信号）为参数，此外它还有一个参数用来指定是否进行了差分编码。

① 实际代码中写在一行，这里为使排版美观有自动折行。

预编码模块的信号处理操作在work函数中实现，其核心是一个状态机，共有两个状态：预编码阶段 (STATE_PRECODE) 和导频调整阶段 (STATE_PILOT_ADJ)。初始时处在预编码阶段，将输入数据除以信道估计值（信道估计输入或者 1），同时将最近的输入转换为二进制与导频接入码做比较，如果匹配（错误的位数小于给定阈值）则认为找到了导频同步头，进入导频调整阶段，同时将标志输出置 1。在导频调整阶段，我们将调制了的导频同步头和导频信号调整为复数 0/1 值。当把这段长度的导频调整完成后，又会回到预编码阶段，重新开始对用户数据进行预编码。图 4.5 给出了预编码模块工作过程中状态的转换关系。

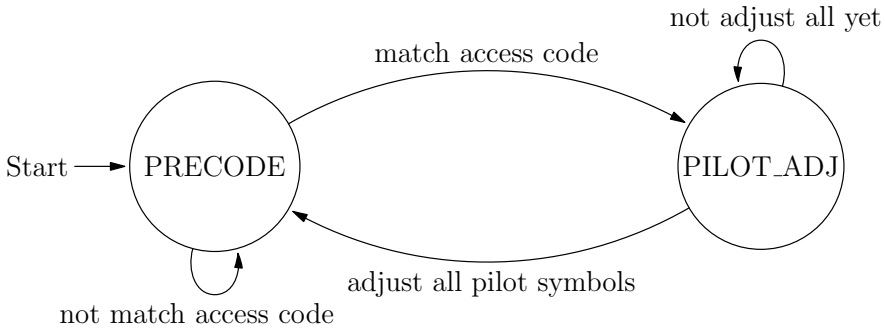

图 4.5　预编码模块状态转换图

注意到信道估计值可能出现幅度很小的值，甚至是零。将输入数据除以这些值会出现异常的输出，这在实际系统中可能会很危险。为了解决这个问题，我们设置了一个变量 d_h_min作为门限，当遇到幅度低于该门限的信道估计值时，我们会直接将输入数据送到输出。这个门限可以通过类的成员函数 set_h_min 由用户根据实际场景设定。这点考虑有助于提高我们系统的可靠性和灵活性。

4.2.4　系统结构

加入了信道估计模块的接收机结构如图 4.6。因为我们不在接收端抵消信道影响，所以信道估计和反馈作为接收机通路的一个旁路。信道估计模块前端的 RRC 滤波器用于匹配滤波，同时它对信号进行位定时恢复，并把数据速率调整到每比特一个样点。由于我们需要信道的幅度信息，所以它需要接在 AGC 之前。

图 4.7显示了加入了预编码模块的发射机结构。由于预编码需要在星座点上

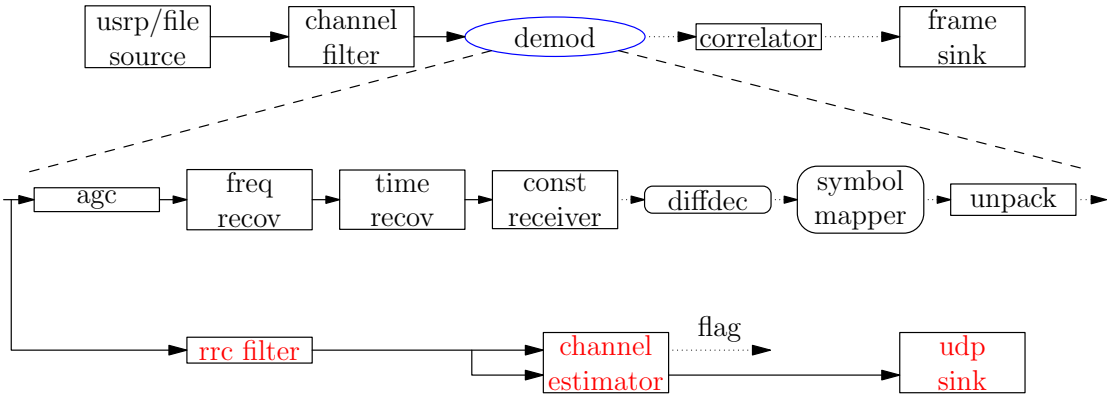

图 4.6　单发单收系统中接收机结构框图

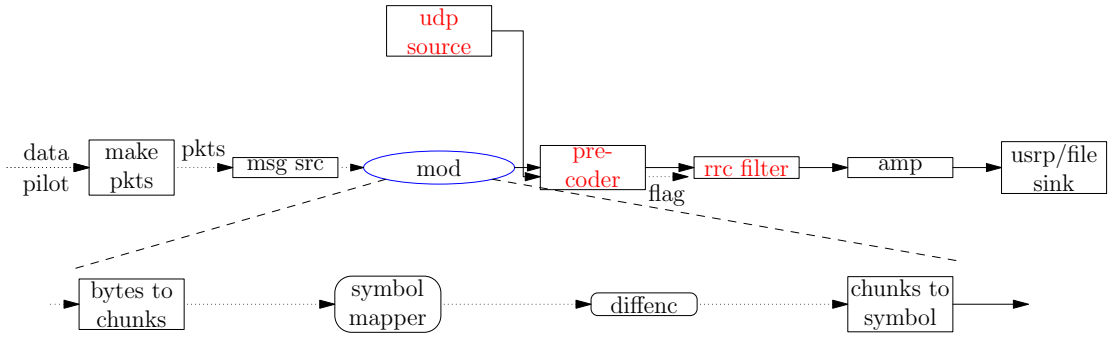

图 4.7　单发单收系统中发射机结构框图

操作，但又需要在 RRC 滤波之前，所以我们修改了通用调制模块，将 RRC 滤波移到了上层发射机通路中，并在其前面加入了预编码模块。注意预编码模块对于发射机通路是必需的，但预编码算法可以选择是否执行（不接入预编码模块的第二个输入端即可不进行预编码）。这样一个系统可以用来对比分析预编码算法带来的性能提升。

为了能够用软件模拟的方式验证算法，我们使用 GNU Radio 模块搭建了一个仿真信道。它可以用来模拟瑞利块衰落信道。在实现上，我们首先定义了一个函数`rayleigh`，它可以返回一个或多个独立同分布的复高斯变量：

```
def rayleigh(rndm, N=1):
    """Generate rayleigh fading channel for
       simulation.

    :param rndm: random.Random object
    :param N: length of output
```

```
    :return a complex scalar if N = 1, or a list of
       length N.

    """
    h = [rndm.normalvariate(0,1/math.sqrt(2)) +
         1j * rndm.normalvariate(0,1.0/math.sqrt(2))
            for i in xrange(N)]
    if N == 1:
         return h[0]
    else:
         return h
```

然后我们只需每隔一段时间调用这个函数，以新的随机数作为新的信道值，这可以通过一个线程实现。模块实现中，我们用一个复常数乘法器模拟瑞利信道的作用，复常数会被周期性更新。我们用gr_channel_model模拟加性高斯白噪声。整个仿真信道的结构如图 4.8所示。

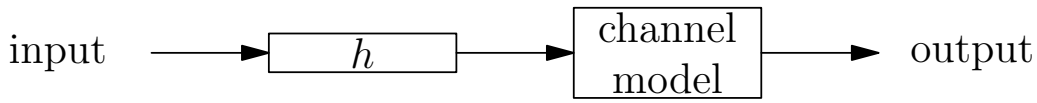

图 4.8　单发单收系统仿真信道结构框图

4.3　多天线系统

4.3.1　信道估计

根据第2.2节描述的多天线系统中的信道估计问题，我们可以知道发送端每根天线同时发送导频，接收端的每根天线可以根据收到的序列估计出各个发射天线到该接收天线的信道值，对应信道矩阵 **H** 的一行。

我们实现的多天线系统中的信道估计模块niulab_chn_est2_cc 构造时的参数如表 4.1所示。第一个参数指定信道估计算法类型，用于选择不同的信道估计算法；第二个参数为导频同步头，用来辅助信道估计模块确定导频位置，解决接收端和发送端之间不同步的问题；最后一个参数给出所有导频的列表，

相当于数学上的导频矩阵 **P**，实现中在 C++ 中为向量的向量，在 Python 中对应列表的列表或者元组的元组[①]，由于向量的大小直到运行时才确定，所以信道估计模块可以应用于任意正整数根发射天线的场景。[②]

表 4.1　多天线系统信道估计模块参数表

类型	形参名
`chn_est2_type_t`	`type`
`const std::vector< unsigned char > &`	`sync_code`
`std::vector< std::vector< unsigned char > >`	`pilots`

信道估计模块的输入端口只有一个，即接收到的经过 RRC 滤波后的星座点数据。借助于一个能够同时进行 RRC 滤波和位定时恢复的模块，我们没有必要将输入分为幅度输入和相位输入两个。模块输出端口数应为导频数加 1，其中第一路为标志输出，其余依从为信道矩阵 **H** 相应行的估计值。由于导频数只有到运行时才能确定，所以我们在运行时检查输出端口数与导频数的上述关系是否成立，如果不成立则抛出`std::runtime_error`异常。

系统的主要工作在`work`函数中进行，其主体实现了一个两状态的状态机，状态切换过程和单发单收系统的类似，参见图 4.4。显著的不同点在于每次需要输出多路估计值。因为导频数目到运行时才知道，我们使用指针的指针来取出多路输出的基地址：

```
gr_complex **out_h =
        (gr_complex **) &output_items[1];
```

然后用`out_h[j]`,$j=0,1,\cdots,N_{\text{pilot}}-1$ 取出每路信道估计输出，`out_h[j][i]`指定输出中的当前样点。另外一点是计算信道估计值前需要先计算出与 **S** 相乘的矩阵。对于目前实现的最小二乘算法，它就是 **P** 的伪逆，给定导频矩阵后它就是固定值，可以在初始化阶段计算得到。我们用一个二维向量`d_adj`保存这个矩阵的转置（行数等于导频数，列数等于单个导频的长度），这样计算信道估计就是计算输入信号与`d_adj`每行的点积，实际实现中我们使用逐点相乘累加得到。

① SWIG 将 C++ 中的向量对应为 Python 中的元组，但传入列表参数也是可以的。Python 中元组是不可变的 (immutable)而列表是可变的。

② 当然实际接收系统有内存限制，实际发射天线数也不会特别大。

4.3.2 反馈信道

多天线系统中的反馈信道实现上，我们仍然采用 UDP 数据包的方式。与单发单收系统的不同点在于，我们需要多个 UDP 端口接收信道估计值。为此我们使用 Python 列表创建一组 gr_udp_sink的实例：

```
self.chn_udp_rx = []
for x in udp_dst:
    [host, port] = x.split(":")
    port = int(port)
    self.chn_udp_rx.append(gr.udp_sink(
        gr.sizeof_gr_complex, host, port))
```

它们中的一个可以通过类似数组下标的方式引用。

发送端使用类似的代码创建出一组 gr_udp_source的实例，以获取信道估计值供预编码模块使用。具体的代码实现在transmit_path.py 的 transmit_path2类中，这里不再列出。

4.3.3 预编码

我们进一步实现了多天线系统中 2×2 场景下的预编码算法模块 niulab_precoding2_cc。它的构造参数与单发单收系统的相同，都为导频接入码、导频总长度和调制器是否使用差分编码。输入输出各有两路，分别为预编码前后两个待发送的数据流，另外还有四路信道估计值输入和一路预编码标志输出。

模块的work函数的核心仍然是个两状态的状态机，状态切换过程和单发单收系统的类似，参见图 4.5。主要的不同点在于预编码之前需要计算得到预编码矩阵。因为信道估计值作为输入，可能随时发生变化，所以每次都要更新预编码矩阵。然而对于瑞利块衰落的场景，信道值本不会很频繁地变动。针对这个特点，我们使用一个四元素数组d_H_last记录之前的信道估计值。只有当前的四个估计值与过去的信道估计值的差值的幅度大于阈值d_H_diff_threshold时才真正重新计算预编码矩阵，这样比起逐点计算大大降低了开销。

在具体的预编码算法上，我们已实现的是迫零算法。在 2×2 场景下，信道矩阵 **H** 的伪逆等同于逆：当 **H** 满秩时，二者存在且相等；当 **H** 不满秩时，二

者都不存在。而 2×2 矩阵的逆有着比较简单的结构：

$$\mathbf{H}^{-1}=\begin{bmatrix}a & b\\ c & d\end{bmatrix}^{-1}=\frac{1}{\det(\mathbf{H})}\begin{bmatrix}d & -b\\ -c & a\end{bmatrix}=\frac{1}{ad-bc}\begin{bmatrix}d & -b\\ -c & a\end{bmatrix} \tag{4-1}$$

所以预编码矩阵可以容易得到。注意当 **H** 的行列式幅值很小时，信道矩阵的行接近线性相关，逆矩阵将给出异常结果。实现中我们会忽略这次的信道估计值，不更新预编码矩阵。

4.3.4 系统结构

多天线系统中的接收机结构如图 4.9所示，结构与单发单收系统相比主要差别在于信道估计模块输出多个信道估计值，对应 UDP 末端也有多个（图中画了两个作为示意）。代码中接收机实现为 receive_path.py 中的 `receive_path2` 类。

发射机结构如图 4.10所示，其中未画出调制模块的内部结构（和图 4.7中的相同）。这个发射机可以对应单个小区基站多天线的场景，也可以对应两个小区每个基站处一根天线的基站协作场景。对于后者来说，每个基站各有一个如图 4.7的发射机，预编码模块的非本小区数据输出弃置不用。发射机实现在 transmit_path.py 中的 `transmit_path2` 中，注意这个类中不含最后的 USRP 或文件末端，这样可以支持后面接上仿真信道进行软件模拟。预编码模块的四个信道估计值输入在图中用粗线表示，它们分别对应接收机反馈回来的信道估计结果。

2×2 场景下的仿真信道在 channel_path.py 中实现，它用四个常数乘法器和两个加法器模拟信道矩阵 **H** 的作用，两个`gr_channel_model`模块模拟加性高斯白噪声。仿真信道的结构框图如图 4.11所示。

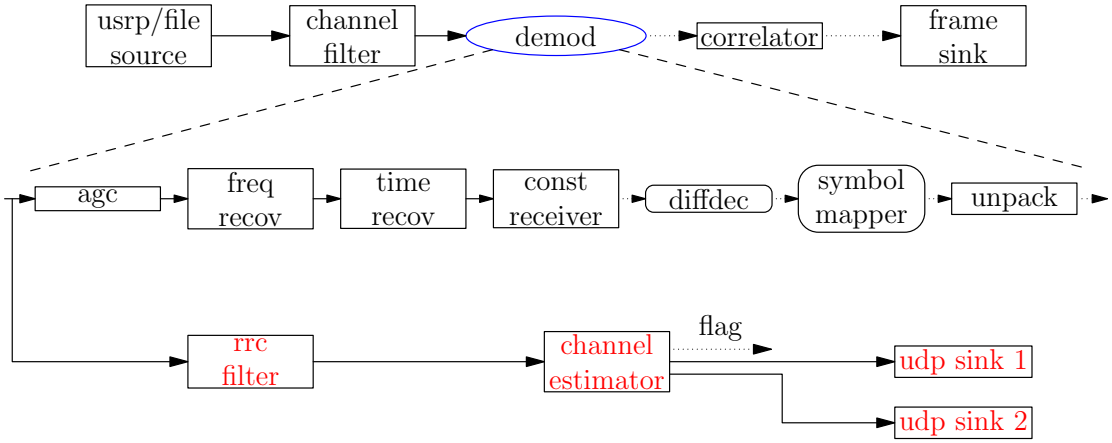

图 4.9　多天线系统中接收机结构框图

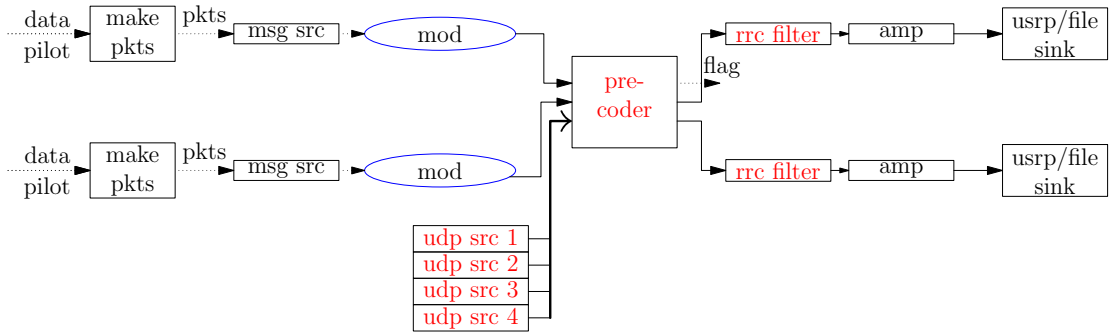

图 4.10　多天线系统中发射机结构框图

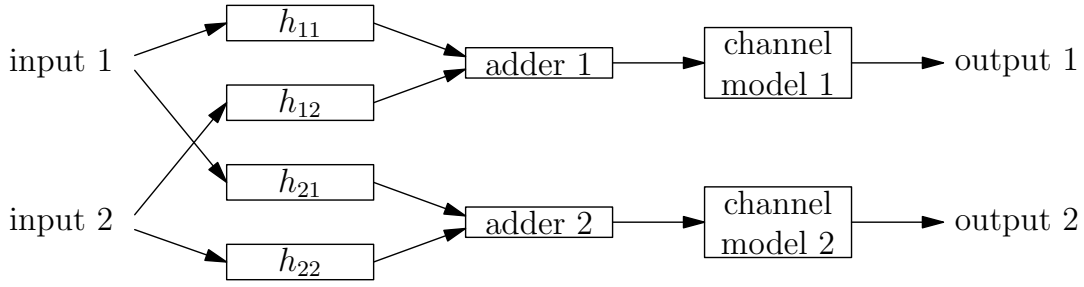

图 4.11　多天线系统仿真信道结构框图

第 5 章　测试结果与分析

我们将设计实现了的带有预编码的单发单收系统和多天线系统分别在软件模拟的瑞利信道和真实的无线环境中做了测试，测试结果和相关分析将在下面介绍。

5.1　软件测试

5.1.1　单发单收系统

我们编写了 `bm_precoder_1x1.py` 脚本以对实现了信道估计和预编码算法的单发单收系统进行软件测试。这个脚本实现了一个包含发射机、接收机和中间信道的完整系统，框图如图 5.1。发射机使用了 `niulab_precoding_cc` 模块做预编码，接收机使用 `digital_mpsk_chn_est_cc` 模块做信道估计，中间信道使用乘法器和 `gr_channel_model` 模块模拟有瑞利衰落和加性高斯白噪声的无线信道。我们利用一个线程每隔5 s生成一个新的瑞利分布的随机数来模拟块衰落的瑞利信道。

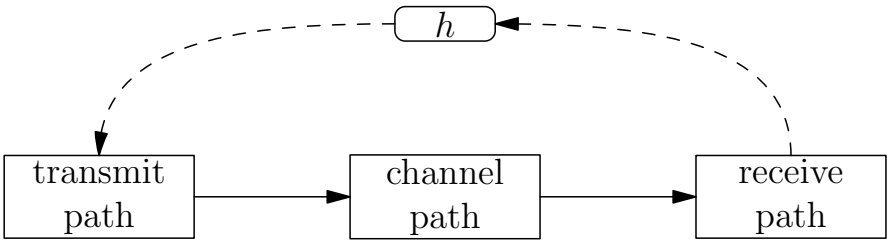

图 5.1　单发单收系统结构示意图

我们在测试中每发送 5 个用户数据的数据包后，发送一个导频信号的数据包以进行训练。系统调制解调方式是非差分的 BPSK，使用的接入码、导频接入码、导频同步头、导频等参数见表 5.1。接收机会每隔 0.5 s 向文件 `rx_chn_est.log` 输出当前的信道估计值、接收信号频偏、位定时偏移等信息。当接收机接收到数据包后，会在日志文件 `rx_pkt.log` 中记录数据包编号、校验是否正确以及已经接收到的数据包数目和正确的数据包数目。我们的测试脚本还负责读取命令行参数，生成一定量的数据打包发射。我们可以通过

命令行参数指定发包数、数据包长度、信道 SNR 值、是否进行预编码、是否记录模块数据到文件以做后续分析等等。图 5.2是持续更新的测试过程的一个截图，显示了若干有用的输出信息。

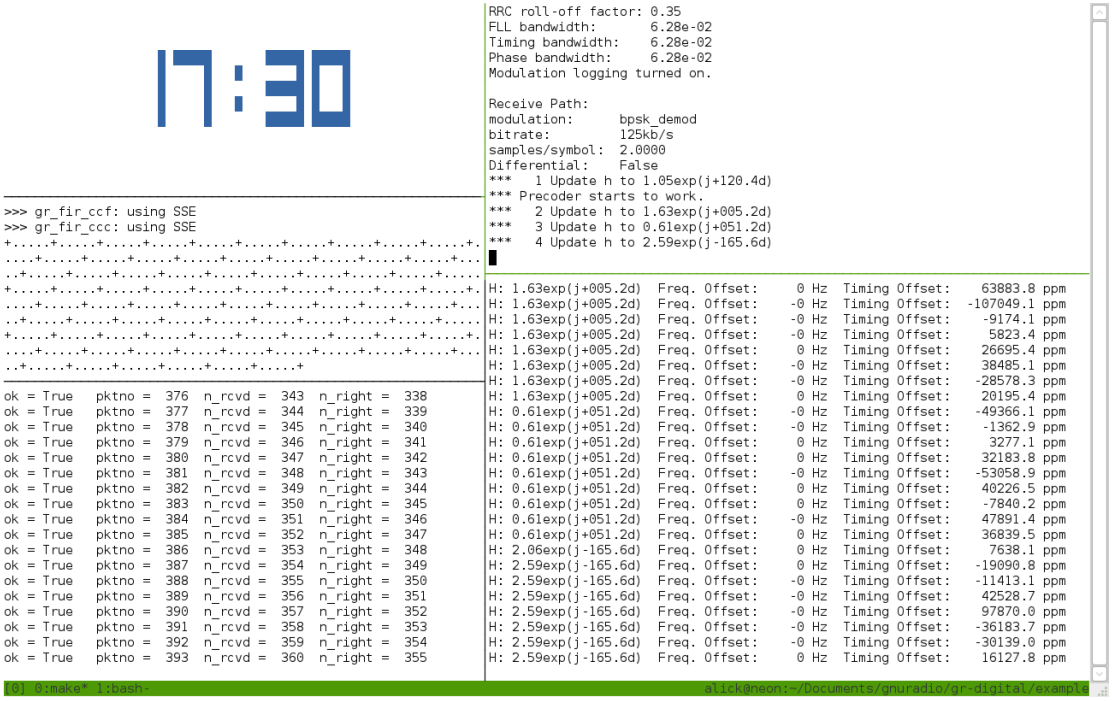

图 5.2　单发单收系统实时测试平台截图。图中将窗口分成为五个窗格。右上方是启动程序的窗格，同时输出各模块的配置和仿真信道的更新；右下方显示信道估计等信息；左下方显示接收到的数据包统计信息；左边中间窗格显示发包情况，一个点代表一个用户数据包，一个加号代表一个训练序列数据包；左上方是一个时钟。

表 5.1　单发单收系统测试的同步码和导频

名称	码字（十六进制记法）
接入码	ACDDA4E2F28C20FC
导频接入码	BD4B3D14344BA151
导频同步头	0000FFFFFFFF0000
导频	FFFFFFFFFFFFFFFF

要使测试系统可以运行，我们需要解决一个“鸡生蛋蛋生鸡”的问题：如果直接将信道估计的结果送到 UDP 端口，预编码模块直接从该端口取信道估计值的话，这些模块将始终得不到输入数据从而无法输出数据。我们的解决方法

是一开始时向 UDP 端口发送全部为 1 （复数）的数据，并使用一个线程在启动 1 s 后不再发 1，而是将反馈的信道信息发到 UDP 端口。这时真正开始对输入数据进行预编码。

我们可以为仿真信道指定随机数种子，以得到可以重复的结果。当指定瑞利信道随机数种子为 2048 时，信道 h 依次变为 $1.05e^{j120.4°}, 1.63e^{j5.2°}, 0.61e^{j51.2°}$。此时在 SNR 分别为 100 dB 和 20 dB 下的信道估计结果如图 5.3 和图 5.4所示。可以看到估计结果和实际值基本一致，当 SNR 较低时信道估计结果误差会增大。这验证了我们的信道估计算法的正确性。

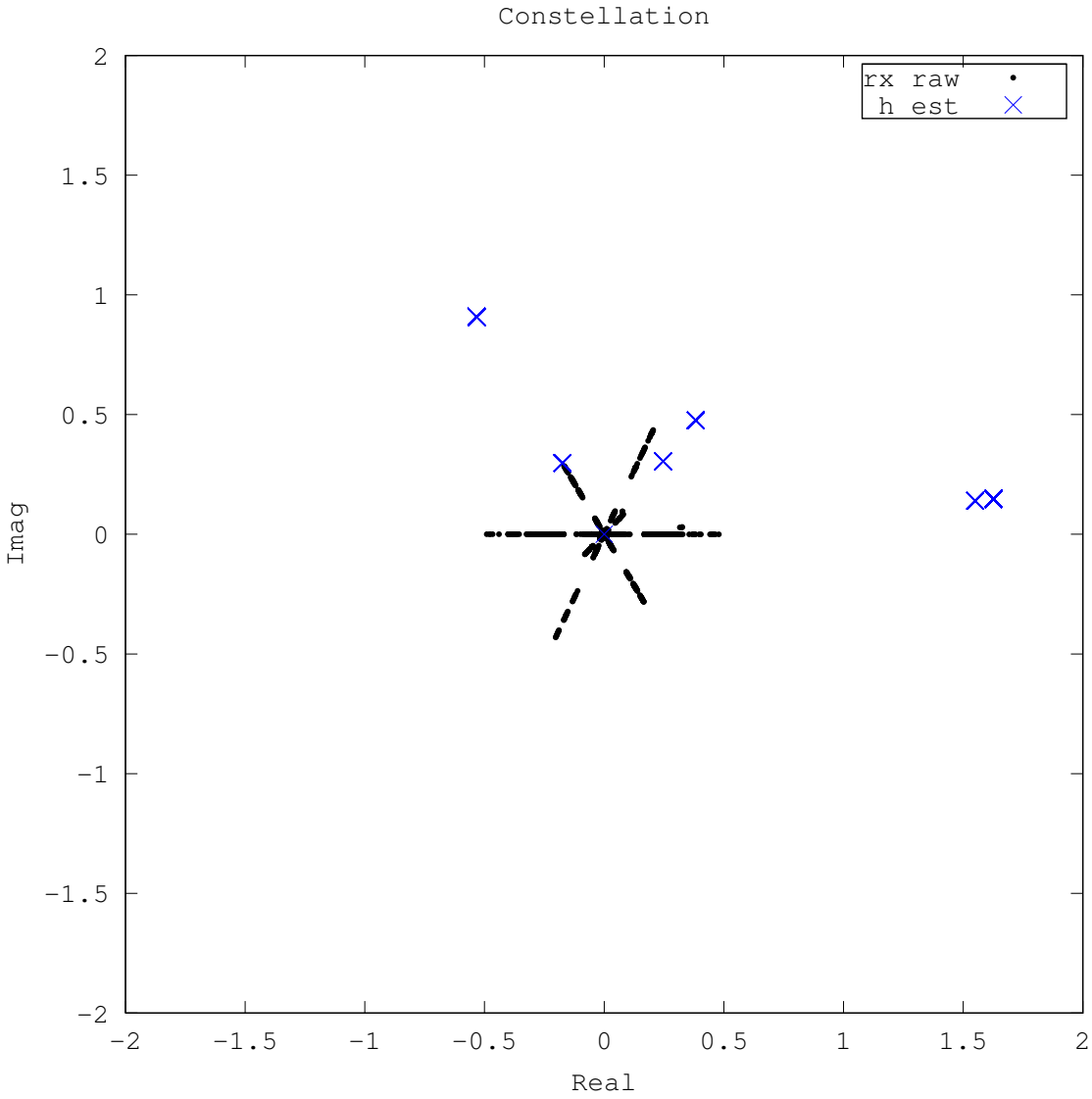

图 5.3　单发单收系统在 SNR= 100 dB 时的信道估计结果。图中 rx raw 指接收机最前端的接收信号，h est 为信道估计结果。

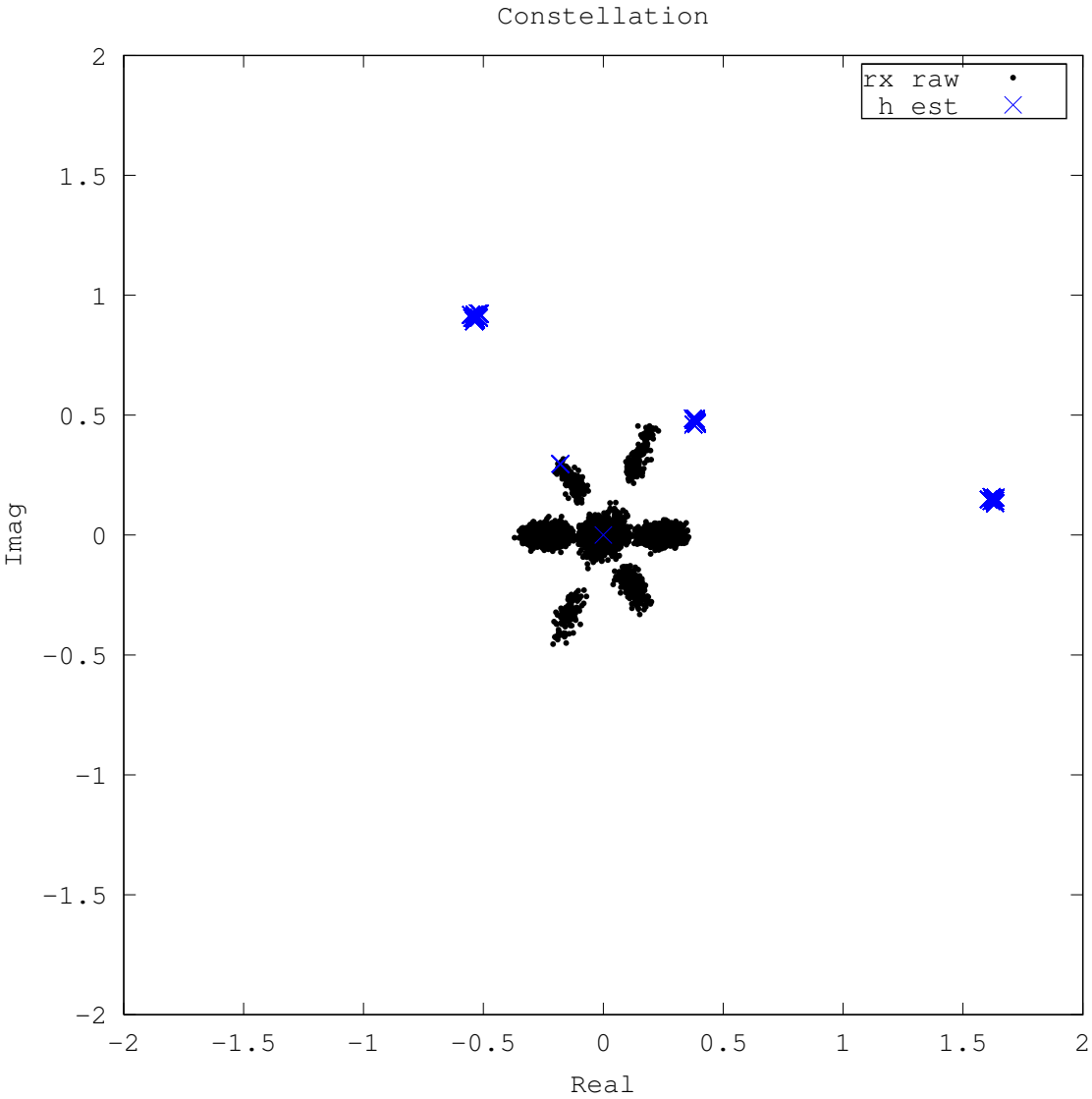

图 5.4　单发单收系统在 SNR= 20 dB 时的信道估计结果。

取定 SNR= 100 dB，使用预编码前后一个数据包的部分数据在发送端和接收端星座图上的分布如图 5.5 和图 5.6 所示。此时信道值为 $h = 1.05\mathrm{e}^{\mathrm{j}120.4°}$，可以看到不使用预编码时接收到的符号的星座点旋转了约 120°，在 BPSK 调制方式下将会导致解调结果恰好与原始数据相反。而当采用了预编码后，发送符号在通过预编码模块后预先旋转了 $-120°$，这样到了接收端时恰好恢复到了正确的位置，数据得以正确解调。预编码算法的正确性和带来的性能提升得以验证。

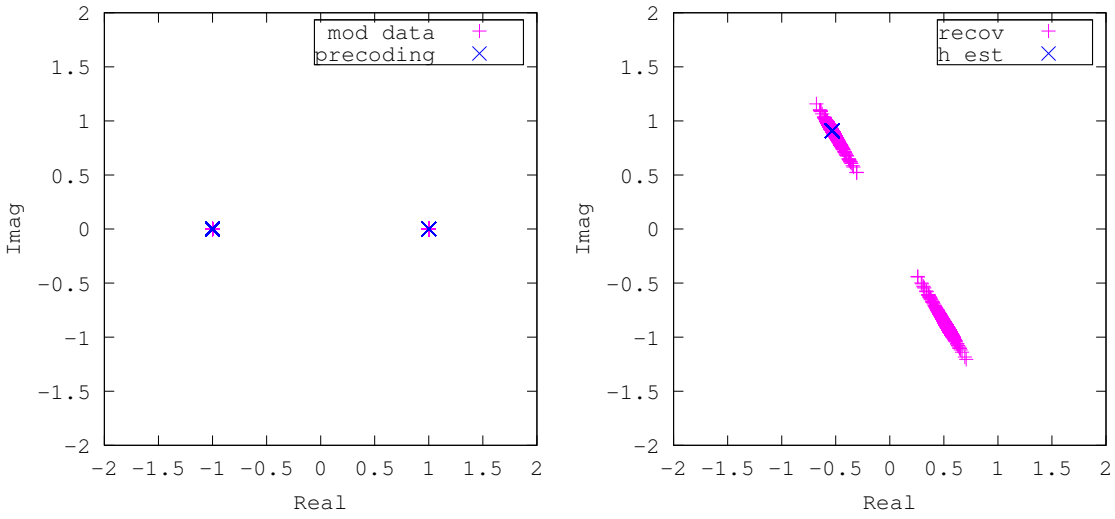

图 5.5　单发单收系统不使用预编码时收发星座图。左子图是发送端已调数据及其通过预编码模块后的星座点，右子图显示了接收端接收数据经过位定时恢复后判决之前的星座点和信道估计值。不使用预编码时，预编码模块对用户数据不做任何处理，只是把输入送到输出 (pass through)。

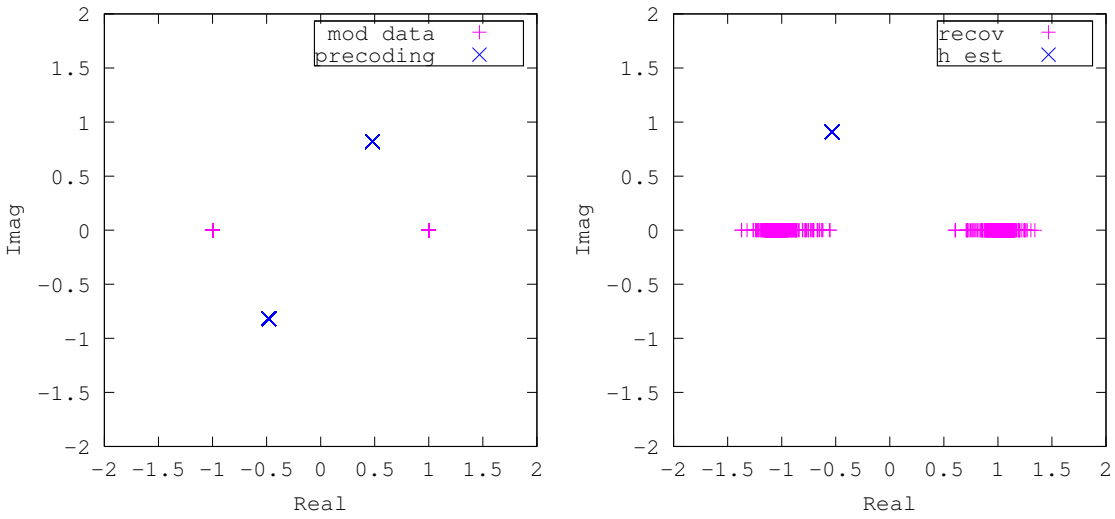

图 5.6　单发单收系统使用预编码时收发星座图。各数据点含义与上面相同。当使用预编码时，预编码模块会对已调数据进行预编码，抵消信道影响。

图 5.7　单发单收系统不使用预编码时接收图片数据

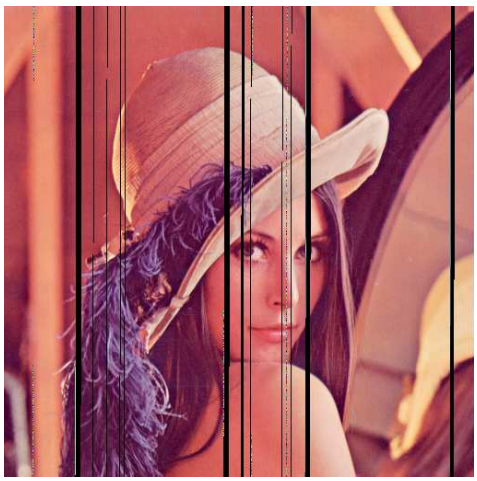

图 5.8　单发单收系统使用预编码时接收图片数据

我们还通过收发图片数据的方式测试了单发单收系统的性能。预编码前后接收到的图片数据恢复出的图片分别如图 5.7 和图 5.8所示。可以看到不使用预编码时只接收到约一半的数据，而使用预编码后可以接收到绝大多数数据。测试是在 SNR= 100 dB 条件下使用 BPSK 调制方式进行，这时接收信号主要受衰落信道的影响。因为瑞利衰落中 h 相位在 $[-\pi,\pi)$ 上成均匀分布，所以没有预编码时会有一半的数据点有超过180°的相偏，使得解调结果出错。而使用预编码后，对于绝大多数数据点，信道的影响被预编码抵消，因此可以被正确解调。

5.1.2 多天线系统

我们对多天线系统在 2×2 的配置下进行了测试。为此我们编写了脚本bm_precoder.py，它实现了带有两个发射机（在同一个类中）、两个接收机和一个模拟的瑞利信道的顶层模块，对应的系统框图如图 5.9所示。

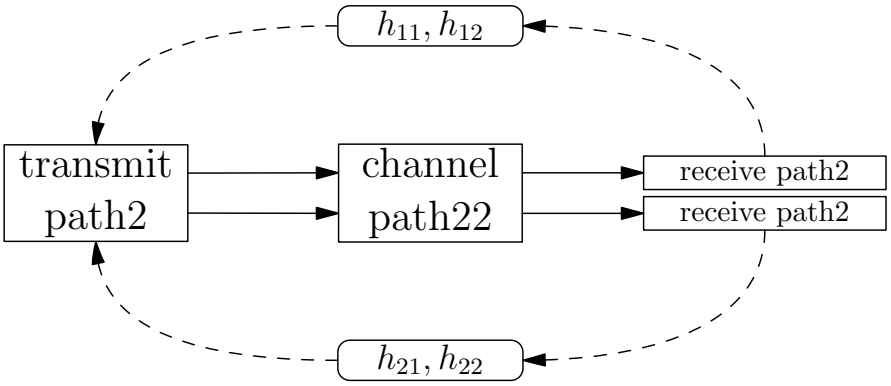

图 5.9 多天线系统结构示意图

脚本接收的命令行参数、工作流程与单发单收系统的测试脚本类似。与其不同的是，接收机接收数据包的日志文件 rx_pkt.log 中增加记录接收机的编号以区分两个接收端。接收机的信道估计信息日志 rx_chn_est.log 中记录两个接收机估计得到的信道值（4 个复数）。我们使用一个线程在启动 0.2 s 后将反馈的信道信息接入到预编码模块真正开始预编码，使用另一个线程每隔 10 s 更新一次信道矩阵 **H** 的值。我们设定在两个发射机各发送五个数据包后同时各发送一个导频帧以测量信道更新信道估计值，调制解调方式仍然是 BPSK。测试时使用的接入码、导频接入码、导频同步头、导频等参数见表 5.2。图 5.10显示了多天线系统测试中的输入输出界面。

表 5.2 多天线系统测试的同步码和训练序列

名称	码字（十六进制记法）
第一路接入码	ACDDA4E2F28C20FC
第二路接入码	AD435C08636F987B
导频接入码	BD4B3D14344BA151
导频同步头	0000FFFFFFFF0000
第一路导频	0000FFFF0000FFFF
第二路导频	FFFF0F0FFFFF0F0F

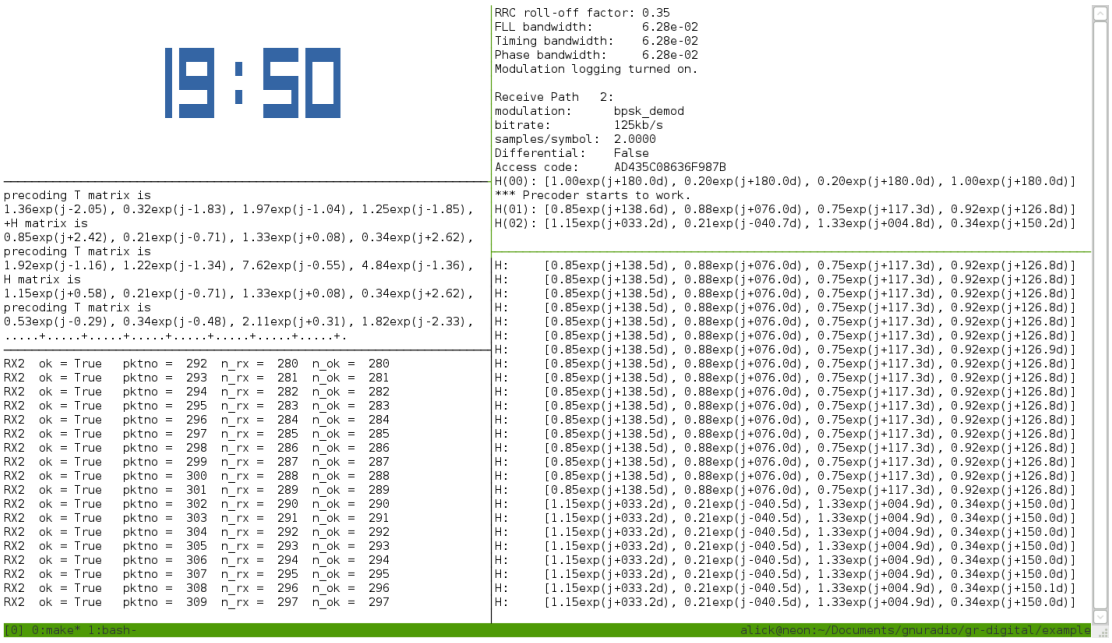

图 5.10　多天线系统实时测试平台截图。各窗格功能和单发单收系统的类似。不同在于输出的内容格式上。另外预编码矩阵有更新时也会将新的预编码矩阵输出到左边中间的窗格中。

当信道随机数种子设为 525 时，最先生成的瑞利信道为

$$\mathbf{H}=\begin{bmatrix}0.56\mathrm{e}^{\mathrm{j}106.3^\circ} & 0.62\mathrm{e}^{\mathrm{j}-165.2^\circ}\\ 0.64\mathrm{e}^{\mathrm{j}-96.9^\circ} & 0.87\mathrm{e}^{\mathrm{j}145.1^\circ}\end{bmatrix}.$$

给定 SNR=50 dB，使用预编码前后两路发送和接收的星座点图如图 5.11–5.14所示。首先可以看出接收端信道估计结果和 $\mathbf{H}$ 基本相同，信道估计比较准确。当不使用预编码时，预编码模块就是一个直通模块，相当于预编码矩阵为全等阵 $\mathbf{I}_{2x2}$。可以看到此时接收端恢复出的信号偏离了 ±1 的位置。当使用预编码时，根据反馈信息计算出的预编码矩阵为

$$\mathbf{T}=\begin{bmatrix}1.01\mathrm{e}^{\mathrm{j}-118.1^\circ} & 0.72\mathrm{e}^{\mathrm{j}111.5^\circ}\\ 0.74\mathrm{e}^{\mathrm{j}179.8^\circ} & 0.65\mathrm{e}^{\mathrm{j}-156.9^\circ}\end{bmatrix}.$$

可以验算 $\mathbf{HT}=\mathbf{I}$。图 5.13 和图 5.14显示预编码后接收端恢复出的信号回到了 ±1 的位置，这验证了预编码算法的正确性。

我们不设定信道随机数种子多次运行脚本，统计了不同的随机信道下数据包传输的成功率。我们每次在每个发射机发送 500 个数据包（不含导频训练的帧），记录每个接收机最终接收到的数据包数和正确的数据包数。我们设定

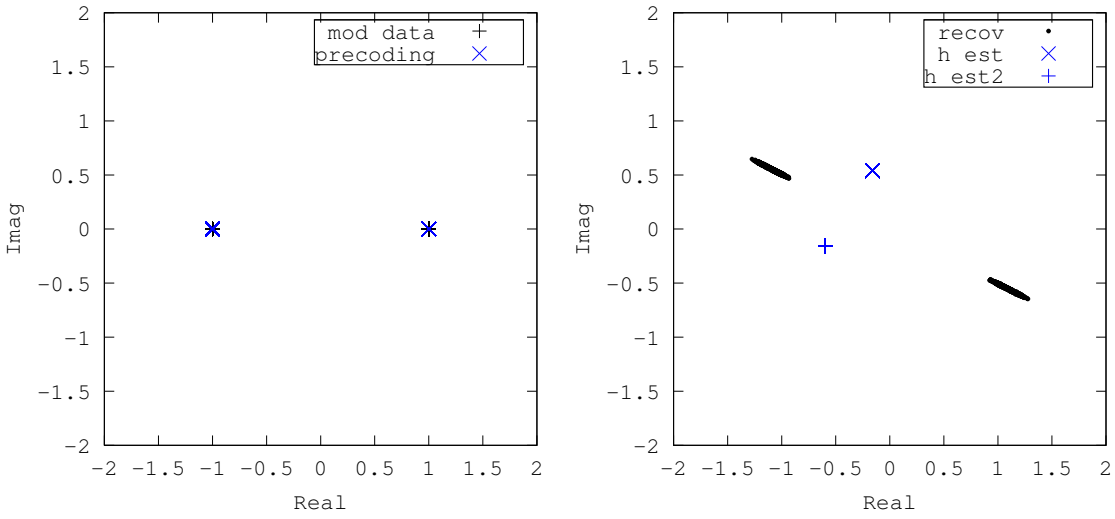

图 5.11　多天线系统不使用预编码时第一路收发星座图。左子图是发送端已调数据及其通过预编码模块后的星座点，右子图显示了接收端接收数据经过位定时恢复后判决之前的星座点和信道估计值 h_{11} 和 h_{12}。不使用预编码时，预编码模块对用户数据不做任何处理，只是把输入送到输出 (pass through)。

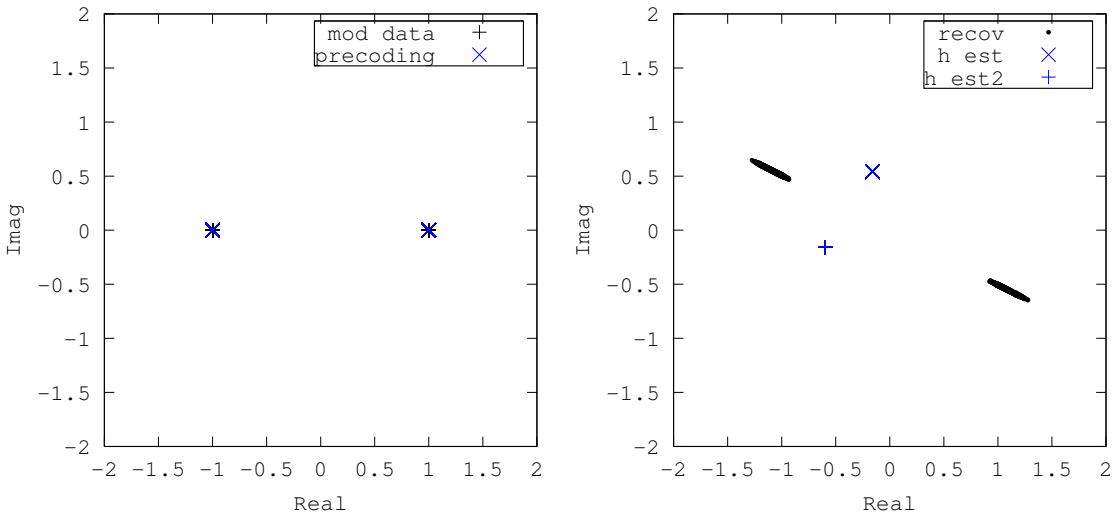

图 5.12　多天线系统不使用预编码时第二路收发星座图。各数据点含义同上，信道估计值为 h_{21} 和 h_{22}。

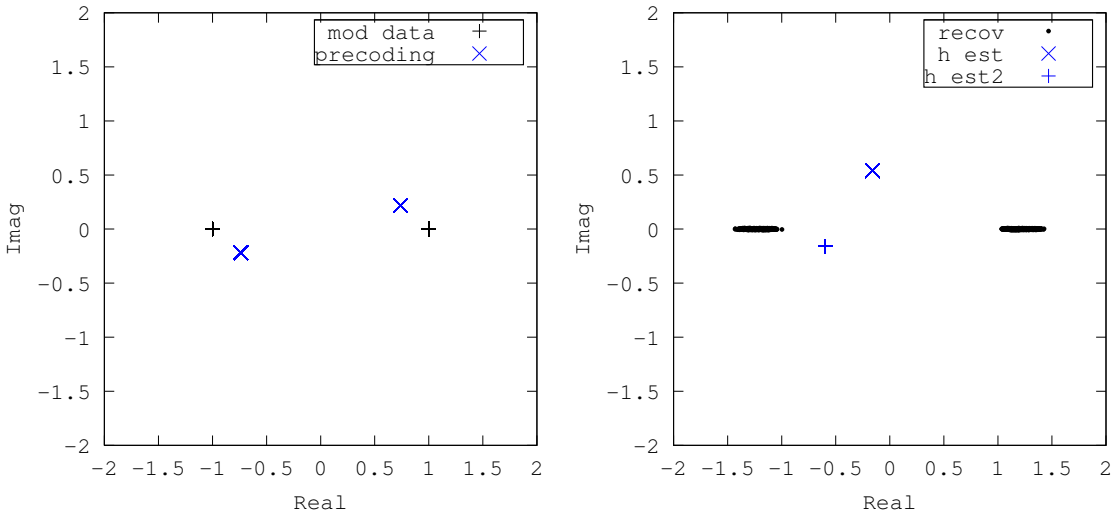

图 5.13　多天线系统使用预编码时第一路收发星座图。各数据点含义与图 5.11中相同。注意这里预编码模块对两路已调数据进行协作预编码，抵消信道 h_{11} 和 h_{12} 的信道影响。

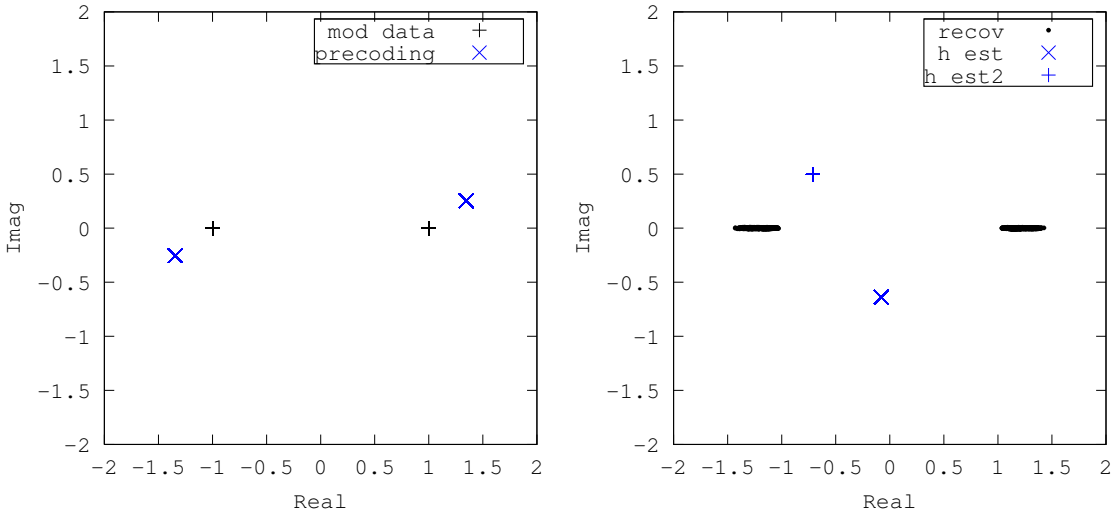

图 5.14　多天线系统使用预编码时第二路收发星座图。各数据点含义与图 5.11中相同。注意这里预编码模块对两路已调数据进行协作预编码，抵消信道 h_{21} 和 h_{22} 的影响。

SNR=100 dB 在使用预编码前后分别记录了 10 次的数据，得到表 5.3 和表 5.4所示的结果。设定 SNR=20 dB 使用预编码前后的结果如表 5.5和表 5.6 所示。表头中 rx1,ok1,rx2,ok2 分别表示第一个接收端接收到的数据包数与正确接收的数据包数和第二个接收端接收到的数据包数与正确接收的数据包数。总计一行是 10 次数据的总和，比例一行是总和除以总发包数 (5000)的结果。可以看到在多天线场景下，不使用预编码时只能接收到很少的数据包，而使用预编码后可以接收到多数数据包（SNR=100 dB时约 92%）。事实上在高 SNR 场景下，少量的数据包丢失主要发生在 **H** 值刚刚改变时。此时系统尚未来得及更新信道估计结果并反馈回预编码模块。当不使用预编码时，接收端会受到较强的其他用户数据的干扰，处于干扰受限状态。当 SNR 较低时，信道估计准确度降低，预编码的有效性也会减弱，丢包率会增大。这一结果验证了预编码算法的正确性，同时表明预编码能够明显地抑制干扰，提升用户的服务质量。与单发单收系统相比，系统在同样的频带资源下传输了双份的数据，印证了多天线系统可以显著提高系统容量。

表 5.3 MIMO 2 × 2 系统在 SNR 为100 dB时不使用预编码的接收到的数据包统计。

序号	rx1	ok1	rx2	ok2
1	199	199	123	123
2	135	134	0	0
3	0	0	125	125
4	0	0	113	113
5	0	0	0	0
6	112	112	0	0
7	125	125	0	0
8	122	122	0	0
9	65	65	65	65
10	118	118	0	0
总计	876	875	426	426
比例	0.1752	0.175	0.0852	0.0852

表 5.4 MIMO 2 × 2 系统在 SNR 为100 dB时使用预编码的接收到的数据包统计

序号	rx1	ok1	rx2	ok2
1	350	350	410	410
2	486	485	375	375
3	487	487	486	485
4	486	486	486	485
5	486	485	490	490
6	371	371	490	490
7	486	485	486	485
8	486	485	400	400
9	486	485	486	485
10	486	485	494	493
总计	4610	4604	4603	4598
比例	0.922	0.9208	0.9206	0.9196

表 5.5　MIMO 2×2 系统在 SNR 为20 dB时不使用预编码的接收到的数据包统计

序号	rx1	ok1	rx2	ok2
1	97	97	132	131
2	83	80	80	80
3	73	73	220	219
4	71	70	83	83
5	247	246	117	116
6	0	0	93	93
7	92	91	122	122
8	133	132	77	77
9	57	56	124	124
10	151	149	201	200
总计	1004	994	1249	1245
比例	0.2008	0.1988	0.2498	0.249

表 5.6　MIMO 2×2 系统在 SNR 为20 dB时使用预编码的接收到的数据包统计

序号	rx1	ok1	rx2	ok2
1	120	120	233	233
2	297	297	273	272
3	204	204	270	270
4	255	255	258	258
5	309	308	283	283
6	228	227	243	242
7	313	313	190	190
8	268	268	265	265
9	215	215	235	235
10	265	265	213	213
总计	2474	2472	2463	2461
比例	0.4948	0.4944	0.4926	0.4922

5.2 硬件测试

我们利用实验室的USRP设备测试了带有预编码的单收单发系统。图5.15中照片显示了我们的测试场景。我们将收发两个USRP设备放在相距约2 m的位置，中间通过放置障碍物以遮挡直射路径。两个USRP通过USB2.0接口连接到一台PC机上（Dell台式机，配置为Intel Core i7-2600 CPU，8核，主频3.40 GHz，内存4 GB）。这样收发两端的程序运行在同一电脑上，UDP数据包在本地环路上传输。我们需要通过设备序列号区分两个USRP。测试中我们使用BPSK调制方式，比特率是125 kb/s，每个符号采样两个点，因此基带采样率为250 kSample/s。我们测试时的接收频率为2.441 GHz，发送频率设置为2.440 993 000 GHz，这是因为实际系统中收发之间总是存在频率偏移，我们需要通过手动修正发送频率的方式补偿它的影响。具体修正值可以通过发送端发送单频信号观察接收端频谱得到。

图5.15　硬件测试场景

图 5.16是系统测试过程中信道估计日志输出的截图，可以看出路径损耗比较显著，收发之间仍有一定频偏，SNR 值较低。表 5.7给出了若干组预编码前后接收端收到的数据包数和正确的数据包数的实测结果，其中我们每组发送 500 个数据包。从结果中可以看出，采用预编码可以带来性能的提升，但提升幅度有限。究其原因，一是频偏带来的影响，二是实际信道 SNR 较低，而我们目前实现的算法在低 SNR 情景下误差较大性能欠佳。

```
H: 0.38e(j-30.97deg)  Freq. Offset:   222 Hz  Timing Offset:    -4293.0 ppm  SNR: 15.4113808836 d
B
H: 0.37e(j132.12deg)  Freq. Offset:   -20 Hz  Timing Offset:     2200.5 ppm  SNR: 15.513176662 dB
H: 0.34e(j80.73deg)  Freq. Offset:   -66 Hz  Timing Offset:    -2421.9 ppm  SNR: 15.1305305736 dB

H: 0.34e(j-174.37deg)  Freq. Offset:   -33 Hz  Timing Offset:    43364.0 ppm  SNR: 14.8091389743
dB
H: 0.35e(j163.87deg)  Freq. Offset:    99 Hz  Timing Offset:      588.9 ppm  SNR: 15.9018688503 d
B
H: 0.35e(j163.87deg)  Freq. Offset:   171 Hz  Timing Offset:    48711.9 ppm  SNR: 15.6544186866 d
B
```

图 5.16　硬件测试信道估计日志输出截图

表 5.7　硬件测试预编码前后正确接收包数与总接收包数

未使用预编码	使用预编码
103/224	197/252
122/234	163/226
135/164	220/258

由于硬件条件的限制，我们无法在多个 URSP 设备间精确同步[①]，所以无法模拟多小区基站协作预编码的场景。我们寄希望于购置中的新设备以测试我们的多天线系统在真实的无线环境下多小区的性能。

① USRP 可以通过修改硬件、使用外部时钟源的方式配置 MIMO，但我们实验室缺少相应资源。

第 6 章　结论与进一步工作

6.1　结论

毕业设计期间，我们首先通过调研了解了现行蜂窝网络体制中存在的问题，以及多天线系统和协作预编码算法的优势与应用前景。我们还调研了已有的软件无线电平台上的多天线系统实现情况，认识到在 GNU Radio 平台上实现带有预编码算法的多天线系统的现实意义，确定了毕业设计的内容。

我们调研了若干预编码算法和基于导频的信道估计算法，熟悉了软件无线电平台 GNU Radio 上已有的系统结构，为算法和系统的设计实现奠定了基础。

在此基础上我们设计了用于信道估计的导频帧结构，实现了信道估计算法和通用的信道估计模块。我们使用 UDP 数据包将估计的信道信息反馈回发送端，设计实现了发送端预编码算法和预编码模块。我们在 GNU Radio 上单发单收系统中实现了预编码，并进一步实现了利用预编码来抑制小区间干扰的多天线系统。我们对单发单收系统和多天线系统在软件模拟信道和真实的无线信道中做了测试，验证了算法的正确性和系统的可靠性。

6.2　进一步工作

实验室正在购置中的新设备 USRP2 会支持使用 MIMO 电缆进行多个设备间的同步，以实现 MIMO 系统。我们希望能在新的硬件设备上测试我们的多天线系统。

现有系统中的反馈方式使用的是有线信道的 UDP 数据包，这和真实的无线系统相差较大。今后我们会考虑实现频分双工 (FDD) 或者时分双工 (TDD)的反馈方式。目前预编码模块将反馈信息作为单独的输入，使得模块没有很好的伸缩性，我们考虑采用 GNU Radio 的 next 分支中的消息 (message)机制来按需反馈信道估计值，降低反馈信息的开销，同时使得预编码模块只需多个数据输入和多个数据输出端口。

我们的信道估计模块和预编码模块提供了实现多种具体算法的接口，可以进一步实现更多的算法。目前的系统没有一个图形界面，我们希望可以利用GRC 提供更友好的用户交互。

插图索引

表格索引

参考文献

[1] Zhang H, Dai H. Design Fundamentals and Interference Mitigation for Cellular Networks. Advances in Wireless Networks: Performance Modelling, Analysis and Enhancement, 2008. 321

[2] GNU Radio Website, accessed February 2012. http://www.gnuradio.org

[3] Foschini G, Gans M. On limits of wireless communications in a fading environment when using multiple antennas. Wireless personal communications, 1998, 6(3):311–335

[4] Bolcskei H. MIMO-OFDM wireless systems: basics, perspectives, and challenges. Wireless Communications, IEEE, 2006, 13(4):31–37

[5] Foschini G, Karakayali K, Valenzuela R. Coordinating multiple antenna cellular networks to achieve enormous spectral efficiency. IEE Proceedings Communications, 2006, 153(4):548–555

[6] Karakayali M, Foschini G, Valenzuela R. Network coordination for spectrally efficient communications in cellular systems. IEEE Wireless Communications, 2006, 13(4):56–61

[7] Liu L, Zhang J, Yu J C, et al. Intercell Interference Coordination through Limited Feedback. International Journal of Digital Multimedia Broadcasting, 2010, 2010

[8] Jiang C, Wang M, Yang C, et al. MIMO Precoding Using Rotating Codebooks. IEEE Transactions on Vehicular Technology, 2011, 60(3):1222–1227

[9] Dahrouj H, Yu W. Coordinated beamforming for the multicell multi-antenna wireless system. IEEE Transactions on Wireless Communications, 2010, 9(5):1748–1759

[10] Biguesh M, Gershman A. Training-based MIMO channel estimation: a study of estimator tradeoffs and optimal training signals. IEEE Transactions on Signal Processing, 2006, 54(3):884–893

[11] Gupta A, Forenza A, Heath Jr R. Rapid MIMO-OFDM software defined radio system prototyping. Proceedings of IEEE Workshop on Signal Processing Systems. IEEE, 2004. 182–187

[12] Li X, Hu W, Yousefi'zadeh H, et al. A case study of a MIMO SDR implementation. Proceedings of IEEE Military Communications Conference. IEEE, 2008. 1–7

[13] Kaszuba A, Checinski R, Lopatka J. MIMO implementation with Alamouti coding using USRP2. Proceedings of Progress In Electronmagnetics Research Symposium Proceedings, 2011. 899–902

[14] 王晓磊. USRP+GNU Radio 平台下中继信道的设计与实现 [D]. 北京: 清华大学电子工程系, 2009

[15] 姜之源. 协作通信中的速率自适应算法的设计与实现 [D]. 北京: 清华大学电子工程系, 2010

[16] 郭雪莹. 能效优先的中继选择算法设计与实现 [D]. 北京: 清华大学电子工程系, 2011

[17] Ettus Research Website, accessed April 2012. http://www.ettus.com

[18] Leech M. build-gnuradio, accessed May 2012. http://www.sbrac.org/files/build-gnuradio

[19] Braun M. mbant/gr-modtool, accessed May 2012. https://github.com/mbant/gr-modtool

致　谢

感谢清华大学电子工程系对我的培养。

衷心感谢牛志升老师对我毕设的悉心指导，还有陈巍老师在开题和中期检查时给我的建议。

感谢实验室师兄周盛、姜之源、王晓磊、胡聪世、宝雅男、师姐郭雪莹、吴健、张珊等对我的帮助，他们在毕设安排、理论调研和平台熟悉等方面给了我耐心的帮助和指导。

感谢 GNU Radio 邮件列表，特别是其中活跃的 Tom Rondeau, Josh Blum, Marcus D. Leech 等人，他们在我的 GNU Radio 入门阶段为我答疑解惑，指点迷津，使我受益匪浅。

感谢刘景初、杨迎翔、陈爽等同学，我们在毕设期间就各自题目进行了交流，很有收获。

感谢薛瑞尼同学和其他帮助改进 ThuThesis 的同学，毕设论文的 LaTeX 模板让我的论文写作轻松自在了许多，让论文格式规整漂亮了许多。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料的书面翻译

用于多小区多天线无线系统的协作波束成形

摘要

在传统的无线蜂窝系统中，我们基于每个小区做信号处理，小区外的干扰被当作背景噪声。本文考虑了在多天线波束成形系统中协调多个小区中的基站的好处，其中多个基站可以联合优化它们的波束形式以改善整个系统的性能。考虑一个多小区的下行链路场景，其中基站装备有多个发射天线采用线性波束成形或者非线性的脏纸编码，每个远端用户配有单独一个天线，但多个远端用户可以同时在各个小区内活跃。本文关注的设计标准是在满足远端用户信号干扰噪声比 (SINR)限制的前提下最小化总的带权发射功率，或者最小化基站中最大的单位天线功率。本文的主要贡献是一个有效的算法用于找到所有基站中联合全局最优的波束形式。提出的算法以使用拉格朗日对偶理论推广上下行链路的对偶性质到多小区设定为基础。一个重要的特性是它自然地指向时分双工(TDD)系统中的分布式实现。仿真结果显示仅仅协调波束成形向量就已经提供了相比于传统的单小区优化的网络而言可观的性能提升。

关键词：波束成形，脏纸编码，上下行链路对偶性，多输入多输出 (MIMO)，多小区系统

A.1　简介

传统的无线系统设计是一个蜂窝架构，其中不同小区的基站独立地和自己的远程终端通信。信号处理基于单个小区，小区间的干扰被当作背景噪声，小区间的协调仅限于处理移动切换。典型的传统蜂窝网络运行在小区间干扰受限的体制下。因此，如果在不同基站间进行联合的信号处理以最小化甚至消除小区间干扰，，传统网络的性能可以得到显著改善。

本文评估了一种多小区下行链路系统中的特殊类型的基站协作。设计场景中基站配有多个天线，远程接收者每人配有一个天线。在每个小区内，多个远程用户可以同时处于活跃状态，通过使用波束成形进行空分复用将他们分开。

在传统系统中，每个小区的波束成形向量是独立设置的。本文的主要观点是通过协调不同基站的波束成形向量来提升整个网络的性能是可能的。

本文中的系统设置不同于文献（例如 [1], [2], [3], [4]）中研究的许多协调联合处理系统，文献中来自多个基站的天线被作为一个天线阵列。这样的全面协调案例有时候也称为“网络 MIMO”（多输入多输出），可想而知它有大得多的性能收益。然而，网络 MIMO 的代价在于需要信号一级的协调，即不同小区中不同移动用户的数据流需要在基站之间共享。相比之下，本文中考虑的系统只需要在波束成形一级上的协调，因此它更容易实现。

多天线无线系统中的下行链路的波束成形已经在文献中被大量研究。一个称为上下行链路对偶性的概念已经成为主要工具。本文通过论证多小区下行链路波束成形中在满足接收信号干扰噪声比 (SINR)限制的前提下最小化总的带权发射功率，或者最小化基站中最大的单位天线功率的问题可以通过对偶的上行链路问题得到解决，将对偶性扩展到了多小区。对偶性对于线性波束成形和非线性的脏纸编码系统都成立。本文的主要贡献是一个有效的算法，可以找到所有基站的全局最优的下行链路波束向量。这个算法是 [5] 中提出的针对单小区情形的类似算法向多小区的推广。这个算法的关键优点在于它自然地指向时分双工 (TDD)系统中的分布式实现。

除了上面提到的发射功率最小化问题，我们也可以提出在满足基站处功率限制的条件下最大化速率范围的问题。两个问题都有实际意义。尽管显示上下行链路对偶性对于速率范围最大化问题在合适的条件（如所有发射者的功率总和受限）下仍然成立是可能的，数值优化可达到的速率范围变得困难许多。因此，本文限定专注于 SINR 受限条件下的功率最小化问题。

A.1.1 相关工作

蜂窝网络中的基站协作的好处已经是许多近期研究的主题。受到联合检测和合作技术用于小区内干扰抑制的激发，[1], [2], [3], [4], [6], [7], [8] 研究了用于抑制小区间干扰的基站间联合编码或解码带来的容量提升。蜂窝网络中基站配有多个天线在实践中是可以想象的，因为基站是通过高容量的骨干链路连接的。然而，要实现联合处理，骨干上所需的通信量也是很大的。这激发了若干网络模型，或是骨干容量受限的 [9], [10], [11]，或是只在相邻基站之间进行协作 [12], [13], [14] ，或是在基站集群之内协作 [15]。

本文中设定的问题与以上工作中的不同之处在于我们考虑在波束成形一级作协作，而非在信号一级。这样协作需要的开销要少的多，其实现更为实际。特别地，每个用户的数据流只需要在他自己的基站作（预）处理（不是在所有基站之间）。更进一步，各个基站不需要做符号同步，而信号级的协作则需要。通过这些实际的考虑，本文重点在于基站间发射波束成形的联合优化，以在移动用户服务质量的约束下最小化发射功率。这种设置适合有严格延时约束的恒定比特率业务。本文的仿真结果显示，使用这种有限的协调形式就可以得到可观的性能增益。基于最小化丢包率 [16] 和最大化容量 [17],[18] 的相关工作也已经出现在文献中。

发射波束成形设计问题可以回溯到 [19] 中的经典工作，其中提出了一个迭代算法用来在任意的传输链路集合中得到满足一系列目标 SINR 的最优化波束向量和功率分配。[19] 的主要贡献在于一个基于上下行链路对偶性的波束成形 -功率更新算法，它能够收敛到问题的一个可行解。在单小区多用户的情形下，这个基于对偶性的方法的最优性在 [20], [21], [22] 中得到了证明。更近一些，[5] 表明单小区下行链路波束成形问题可以表示成一个二阶锥规划问题。这个关键的洞悉使得可以通过凸优化里的拉格朗日理论解释对偶性 [23]。

如果假定基站间有信号级协作，单小区上下行链路对偶性可以很快的推广到多小区设定下。[24] 在 CDMA 场景中表明了这一点，其中提出了一个与 [19] 中类似的波束成形 -功率迭代算法。

在仅有波束成形一级协作的多小区网络中，上下行链路对偶性是否继续成立的问题没有那么显而易见。本文使用了拉格朗日对偶性方法论证了对偶性在这种情形下的确存在。此外，本文提出了一个仅基于功率迭代的最优化步骤。这个算法有一个关键优势，即可以有分布式实现，这在多小区网络中是很渴望得到的。更进一步，本文考虑了现实的功率限制，解决了所有基站的最大单位天线功率的最小化问题。最后，本文提供了一个推广，在每个小区中包含脏纸编码。

本文中考虑的多小区上下行链路对偶性与 [25] 中提出的网络对偶性概念有关。不过 [25] 中的网络对偶性是通过线性规划的方法派生而来，这不同于本文使用的拉格朗日方法。因此，针对协作波束成形问题 [25] 得到了一系列不同的数值算法，在分布式系统中恰好不容易实现。事实上，要得到分布式方案，[25] 需要诉诸次最优的算法。作为进一步的注解，[25] 处理了多小区网络中更一般

性的联合发射和接收波束成形最优化问题，其中基站和远程用户都配有多个天线。尽管上下行链路对偶性继续有效，这个更一般性的设置的结果是不再能够证明全局最优性，这不同于本文中考虑的简单一些的每个远程用户配有单一天线的情形。最后，协作波束成形问题可以通过另外一种方法求解，这里将所有的基站看作一个发射者，然后修改对应的信道矩阵，确定相应的最优波束向量[26]。不过这样导出的算法还没有分布式的实现。

多小区多天线信道中波束成形级的协作已经被 [27],[28] 从自我主义对利他主义策略的角度进行过探索。这些工作重点在多小区网络在固定功率限制下可以达到的速率范围的巴茉托边界，这和本文中考虑的在固定 SINR 限制下最小化功率的问题恰好互补。多小区系统的可达到的速率方位已经被 [29] 使用对偶性进行了探索，还被 [30] 做了探索。特别地，[30] 处理了一个更一般性的设定，考虑了不同可能等级的基站协作。本文中的问题设定相应于 [30] 中的一个特殊的受限的协作场景。更进一步，[30] 还强调了基站处不完美的信道知识带来的问题。

本文中的问题设定假设每个小区内的活跃移动用户的集合一级他们相应的 SINR 限制都是固定的。用户调度（例如 [31], [32], [33]）和拥塞控制策略假定为单独进行。全文中我们假定每个基站处都有每个小区内的移动用户的完美的信道边信息。在实际实现中，信道边信息可以通过 TDD 系统中的信道互惠或者通过反馈机制来估计，参见 [34], [35]。

A.1.2 组织结构

本文后面的内容如下组织。第二节包含了问题形成和系统模型。第三节中我们论证多小区网络中的上下行链路对偶性用于在 SINR 约束下最小化总的带权发射功率或者最小化最大单位天线功率。第四节包含多小区下行链路波束成形的分布式算法。第五节提供了仿真结果。总结性评注在第六节。

符号记法： $\mathscr{R}$ 和 $\mathscr{C}$ 代表实空间和复空间。单位矩阵记作 I。矩阵的转置和共轭转置分别记为 $(.)^T$ 和 $(.)^H$。

A.2 问题形成

A.2.1 系统模型

本文考虑一个多小区多用户空分复用系统，有 N 个小区，每个小区有 K 个用户，每个基站有 N_t 个天线，每个远程用户一个天线。每个基站都进行多用户下行链路发射波束成形。令复标量 $x_{i,j}$ 表示第 i 个小区中的第 j 个用户的信息信号，$\mathbf{w_{i,j}} \in \mathscr{C}^{N_t \times 1}$ 表示关联到他的波束向量。第 i 个小区中的第 j 个远程用户接收到的信号记为 $y_{i,j} \in \mathscr{C}$，它是要传送的信号、小区内干扰和小区间干扰的加和：

$$y_{i,j} = \sum_{l} \mathbf{h}_{\mathbf{i,i,j}}^{H} \mathbf{w}_{\mathbf{i,l}} x_{i,l} + \sum_{m \neq i,n} \mathbf{h}_{\mathbf{m,i,j}}^{H} \mathbf{w}_{\mathbf{m,n}} x_{m,n} + z_{i,j} \tag{A-1}$$

其中 $\mathbf{h}_{\mathbf{i,i,j}}^{H} \in \mathscr{C}^{N_t \times 1}$ 是第 l 个小区的基站到第 i 个小区中的第 j 个用户的向量信道，$z_{i,j}$ 是实部虚部方差均为 $\sigma^2/2$ 的复的加性循环对称高斯白噪声。图 1 显示了有七个小区、每个小区三个用户的网络的系统模型。

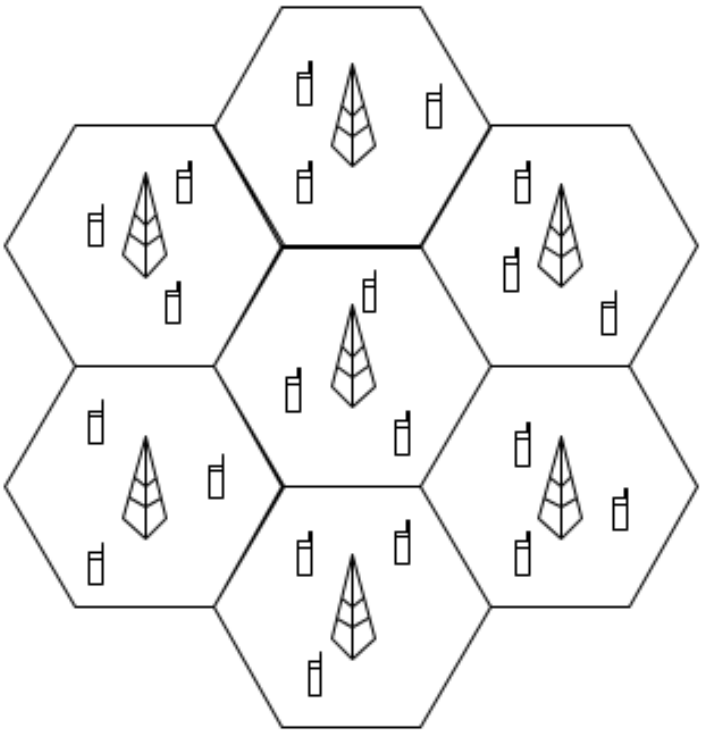

图 A.1　多小区无线网络

A.2.2 发射波束成形问题

波束成形设计问题是在远程用户的 SINR 约束下最小化所有基站的发射功率的某个函数。用 $\mathbf{w}_{\mathbf{i,j}}^H$ 表示波束向量，则第 i 个小区中的第 j 个用户的 SINR 可以表示为：

$$\Gamma_{i,j} = \frac{|\mathbf{w}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2}{\sum_{l \neq j} |\mathbf{w}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2 + \sum_{m \neq i,n} |\mathbf{w}_{\mathbf{m,n}}^H \mathbf{h}_{\mathbf{m,i,j}}|^2 + \sigma^2} \tag{A-2}$$

令 $\gamma_{i,j}$ 代表第 i 个小区中的第 j 个用户的目标 SINR。我们可以示例性的将总的发射功率最小化问题表达如下：

$$\begin{aligned} &\text{minimize} \quad \sum_{i,j} \mathbf{w}_{\mathbf{i,j}}^H \mathbf{w}_{\mathbf{i,j}} \\ &\text{subject to} \quad \Gamma_{i,j} \geq \gamma_{i,j}, \forall i = 1 \cdots N, j = 1 \cdots K \end{aligned} \tag{A-3}$$

其中最小化是针对 $\mathbf{w}_{\mathbf{i,j}}$ 的。全文中我们假定目标 SINR 集合是可行的。

A.2.3 传统系统

在传统的无线蜂窝系统中，多用户波束成形问题是在单个小区的基础上解决的；小区外的干扰被视为背景噪声的一部分。特别地，对于一个固定的基站 $\hat{i}$，传统系统会寻找最优的 $\mathbf{w}_{\hat{\mathbf{i}},\mathbf{j}}, j = 1 \cdots K$ 集合，假定所有其他 $(N-1)K$ 波束成形器都固定：

$$\begin{aligned} &\text{minimize} \quad \sum_{j} \mathbf{w}_{\hat{\mathbf{i}},\mathbf{j}}^H \mathbf{w}_{\hat{\mathbf{i}},\mathbf{j}} \\ &\text{subject to} \quad \Gamma_{\hat{i},j} \geq \gamma_{\hat{i},j}, \forall j = 1 \cdots K \end{aligned} \tag{A-4}$$

其中 $\Gamma_{\hat{i},j}$ 由 (2) 式给出。这个单小区下行链路问题具有经典解法，由 [19], [20], [21], [5] 给出。

注意到在传统系统中，每个基站处的波束成形器的选择会影响到相邻小区的背景噪声的等级，进而影响相邻基站的波束成形器的设定。这样，上面的单个小区最优化在实践中会被迭代地运行直到系统收敛到单小区最优解。

A.2.4 联合优化的动机举例

上面的单小区最优化不一定会指向联合最优解，这一观察激发了本文。如果基站协作起来同时联合最优化所有的波束成形器，则会有显著的性能提升。

下面的例子阐释了这一点。

考虑一个多小区网络，不过每个小区内只有一个用户。单小区优化就化为多输入单输出（MISO）系统的最优发射波束成形问题，其中背景噪声等级包含了小区外的干扰。注意到与背景噪声等级无关，最优的单小区发射波束成形器是一个匹配信道的向量。这样，在这个例子中，所有小区的单小区优化会在一次迭代后收敛 —每个基站都使用匹配 MISO 信道的发射波束成形器。

这个信道匹配解不一定是联合最优的。例如，当属于两个不同小区的两个用户处在小区边缘距离很近时，把两个基站的波束向量指向背离对方以减少相互干扰可能具有优势。这样的联合最优波束成形解可以在固定发射功率的条件下带来更高的接收 SINR，或者相反地，在固定 SINR 条件下使用更低的发射功率。

解决多小区联合波束成形优化问题的最早算法之一由 Rashid-Farrokhi, Liu 和 Tassiulas [19] 给出。他们表明 SINR 约束下的最优下行链路波束向量问题可以通过一个迭代的上行链路波束成形和功率更新算法得到有效解决。众所周知，上行链路波束成形问题要容易解决的多 [36]。如此，通过把下行链路问题变换到上行链路，下行链路问题也可以被有效的解决。

单小区情形中的波束成形 -功率迭代算法的全局最优性已经在 [20], [21], [5] 中被证明。本文给出了多小区情形下对偶形的严格推导，进而提出了一个新算法用于解决联合多小区下行链路波束成形问题。

A.3 多小区系统的上下行对偶性

A.3.1 最小化带权发射功率

我们从用公式表达发射波束成形问题 (3) 的一个稍微更一般性的版本开始。在多小区系统中，每个基站（或者有时是每个天线）经常有其自身的功率约束。因此，考虑最小化带权的总发射功率问题很有用，其中第 i 个基站的发射功率加上权重 α_i。在这个情形下，(3) 变为

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j} \alpha_i \mathbf{w}_{\mathbf{i,j}}^H \mathbf{w}_{\mathbf{i,j}} \\ \text{subject to} \quad & \Gamma_{i,j} \geq \gamma_{i,j}, \forall i = 1 \cdots N, j = 1 \cdots K \end{aligned} \tag{A-5}$$

(5) 中的目标 SINR 约束看起来可能非凸。在一项针对单小区下行链路波束成形问题的研究中，[5] 表明这种类型的 SINR 约束可以变换到一个二阶锥约束（另见 [37]）。这个关键性的观察使得用凸优化解决 (5) 变得可行。

上下行链路对偶性指的是下行链路信道中达到一些特定的 SINR 约束集所需的最小发射功率和上行链路信道中达到同样的 SINR 约束集所需的最小总发射功率一样，这里的上行链路信道通过交换下行链路的输入输出来得到。本节的主要目标就是表明之前论证了的单小区情形下的上下行链路对偶性对于多小区设定同样有效。下面的定理是 [23] 中表述的单小区对偶性到多小区的推广。其证明基于拉格朗日技巧，这和 [23] 中的方法类似。

定理 1： 下行链路多用户多蜂窝网络中的最优发射波束成形问题 (5) 可以通过一个对偶的上行链路信道来解决，其中 SINR 约束保持不变，噪声功率乘上比例因子 α_i。数学上讲，下行链路的最优化问题的拉格朗日对偶是如下的上行链路问题：

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j} \lambda_{i,j}\sigma^2 \\ \text{subject to} \quad & \Lambda_{i,j} \geq \gamma_{i,j} \end{aligned} \tag{A-6}$$

其中最小化是对 $\lambda_{i,j}$ 而言，并且有

$$\Lambda_{i,j} = \max_{\hat{\mathbf{w}}_{\mathbf{i,j}}} \frac{\lambda_{i,j}|\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2}{\sum_{(m,l)\neq(i,j)} \lambda_{m,l}|\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,m,l}}|^2 + \alpha_i||\hat{\mathbf{w}}_{\mathbf{i,j}}||^2}.$$

最优的 $\hat{\mathbf{w}}_{\mathbf{i,j}}$ 可以解释为对偶的上行链路信道的接收方波束向量，它与最优的 $\mathbf{w}_{\mathbf{i,j}}$ 成比例。最优的 $\lambda_{i,j}$ 可以解释为对偶的上行链路功率，它对应于与 (5) 中 SINR 约束相关联的对偶变量。

（由于文章篇幅较长，此处证明过程略去。）

A.3.2 最小化最大天线功率

定理 1 中的权重因子 α_i 提供了一种机制，在多小区网络中不同基站处的功耗之间进行折衷。我们还可以直接地引入另外的权重因子来体现每个天线一级的功耗折衷。但是，选择正确的权重通常不那么容易。不过如果我们考虑一个实际的场景即最小化所有基站中最大的天线功率，那么使用对偶形的进一步扩展，权重可以自动地调整。这种情形下的最优化问题表达如下：

$$\begin{aligned} \text{minimize} \quad & \tau \\ \text{subject to} \quad & \Lambda_{i,j} \geq \gamma_{i,j} \forall i,j \\ & \left[\sum_j \mathbf{w_{i,j}} \mathbf{w_{i,j}^H} \right]_{m,m} \leq \tau, \forall i,m \end{aligned} \tag{A-19}$$

其中 $[\cdot]_{m,n}$ 代表矩阵的第 (m,n) 个元素。下面的定理是 [23] 中单小区每个天线功率最小化问题到多小区的推广。

定理 2： 下行链路多用户多蜂窝网络中的最优的最大天线功率发射波束成形问题 (19) 可以通过一个对偶的上行链路信道来解决，其中 SINR 约束保持不变，噪声不确定。数学上讲，最优化问题 (19) 的拉格朗日对偶是如下的最大 -最小问题：

$$\begin{aligned} \max_{\mathbf{Q_i}} \min_{\lambda_{i,j}} \quad & \sum_{i,j} \lambda_{i,j} \sigma^2 \\ \text{subject to} \quad & \Lambda_{i,j} \geq \gamma_{i,j} \forall i,j \\ & \text{tr}(\mathbf{Q_i}) \leq N_t, \, \mathbf{Q_i} \text{dianonal}, \\ & \mathbf{Q_i} \succeq 0 \forall i \end{aligned} \tag{A-20}$$

其中

$$\Lambda_{i,j} = \max_{\hat{\mathbf{w}}_{\mathbf{i,j}}} \frac{\lambda_{i,j} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h_{i,i,j}}|^2}{\sum_{(m,l) \neq (i,j)} \lambda_{m,l} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h_{i,m,l}}|^2 + \hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{Q_i} \hat{\mathbf{w}}_{\mathbf{i,j}}}.$$

最优的 $\hat{\mathbf{w}}_{\mathbf{i,j}}$ 可以解释为对偶的上行链路信道的接收方波束向量，它与最优的 $\mathbf{w_{i,j}}$ 成比例。最优的 $\lambda_{i,j}$ 可以解释为对偶的上行链路功率，它对应于与 (19) 中 SINR 约束相关联的对偶变量。最优的上行链路噪声协方差阵 $\mathbf{Q_i} = \text{diag}(q_{i,1}, \ldots, q_{i,N_t})$ 是 (19) 中下行链路每个天线的功率约束相关联的对偶变量的对角矩阵。

（由于文章篇幅较长，此处证明过程略去。）

比较 (6) 和 (20)，可以很清楚地看到权重为 α_i 的带权总功率最小化问题对应于 (20) 中的 $\mathbf{Q_i} = \alpha_i \mathbf{I}$。外层对 $\mathbf{Q_i}$ 的最大化提供了一个方法以最优地设置权重以最小化最大天线功率。

A.3.3 使用脏纸编码的波束成形

上下行链路对偶性可以被扩展到允许每个小区内实现脏纸编码 (DPC)。DPC 是一种信息论中的操作，其中下行链路小区内干扰可以在基站处被预先消去。实践中脏纸编码可以用类似 Tomlinson-Harashima 预编码的技术实现。它可以被认为是上行链路中基于接收端的干扰消除的对偶操作。

假设每个小区内预消去的次序特定为 $1,2,\ldots,K$，即第 i 个小区内的第 j 个用户会在基站 i 处通过减去同一个小区内的前 $(j-1)$ 个用户产生的小区内干扰来进行编码。下行链路的 SINR 现在变为

$$\Gamma'_{i,j} = \frac{|\mathbf{w}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2}{\sum_{l>j} |\mathbf{w}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2 + \sum_{m\neq i,n} |\mathbf{w}_{\mathbf{m,n}}^H \mathbf{h}_{\mathbf{m,i,j}}|^2 + \sigma^2}$$

不难发现对偶的上行链路问题和线性波束成形情形中的恰好相同，除了上行链路的 SINR 修改为

$$\Lambda'_{i,j} = \max_{\hat{\mathbf{w}}_{\mathbf{i,j}}} \frac{\lambda_{i,j} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2}{\sum_{(m,l)\succ(i,j)} \lambda_{m,l} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,m,l}}|^2 + \alpha_i ||\hat{\mathbf{w}}_{\mathbf{i,j}}||^2}.$$

其中 $(m,l)\succ(i,j)$ 代表或者 $m>i$，或者 $m=i$ 且 $l>m$。对于最小化最大单天线功率问题也是类似的修改。

A.4 分布式的下行链路波束成形

通过拉格朗日理论推导出上下行链路对偶性构成了下行链路多小区系统中的最优协作波束成形的数值算法的基础。我们的算法基于迭代函数求值的想法，它在 [5] 中考虑单小区情形时被首次提出。本文把这个算法推广到了多小区系统。

多小区系统中算法设计的一个重大考虑是分布式实现问题。在本节的后半部分中我们表明对于 TDD 系统而言，我们提出的算法自然地导出分布式的逐小区实现。

A.4.1 迭代函数求值算法

我们首先展示用于找到带权总功率最小化问题 (5) 的最优波束成形的数值算法。主要的想法是在对偶的上行链路域中解决下行链路波束成形问题，首先

找到最优的 $\lambda_{i,j}$，然后是对应的 $\hat{\mathbf{w}}_{\mathbf{i,j}}$。要找到最优的 $\lambda_{i,j}$，我们首先对拉格朗日函数 (8) 的 $\mathbf{w}_{\mathbf{i,j}}$ 取梯度，并把它置零：

$$\left[\alpha_i \mathbf{I} - \left(1+\frac{1}{\gamma_{i,j}}\right)\lambda_{i,j}\mathbf{h}_{\mathbf{i,i,j}}\mathbf{h}_{\mathbf{i,i,j}}^H + \sum_{m,n}\lambda_{m,n}\mathbf{h}_{\mathbf{i,m,n}}\mathbf{h}_{\mathbf{i,m,n}}^H\right]\mathbf{w}_{\mathbf{i,j}} = 0 \tag{A-26}$$

因此

$$\Sigma_{\mathbf{i}}\mathbf{w}_{\mathbf{i,j}} = \left(1+\frac{1}{\gamma_{i,j}}\right)\lambda_{i,j}\mathbf{h}_{\mathbf{i,i,j}}\mathbf{h}_{\mathbf{i,i,j}}^H\mathbf{w}_{\mathbf{i,j}} \tag{A-27}$$

其中 $\Sigma_{\mathbf{i}}$ 定义在 (11) 式中。

现在，两边同时乘以 $\mathbf{h}_{\mathbf{i,i,j}}^H\Sigma_{\mathbf{i}}^{-1}$，我们得到：

$$\mathbf{h}_{\mathbf{i,i,j}}^H\mathbf{w}_{\mathbf{i,j}} = \left(1+\frac{1}{\gamma_{i,j}}\right)\lambda_{i,j}\mathbf{h}_{\mathbf{i,i,j}}^H\Sigma_{\mathbf{i}}^{-1}\mathbf{h}_{\mathbf{i,i,j}}\mathbf{h}_{\mathbf{i,i,j}}^H\mathbf{w}_{\mathbf{i,j}} \tag{A-28}$$

最后，消去两边的 $\mathbf{h}_{\mathbf{i,i,j}}^H\mathbf{w}_{\mathbf{i,j}}$ 因子，我们得到最优 $\lambda_{i,j}$ 的必要条件：

$$\lambda_{i,j} = \frac{1}{\left(1+\frac{1}{\gamma_{i,j}}\right)\mathbf{h}_{\mathbf{i,i,j}}^H\Sigma_{\mathbf{i}}^{-1}\mathbf{h}_{\mathbf{i,i,j}}} \tag{A-29}$$

这可以被用来迭代地获得最优 $\lambda_{i,j}$。

算法总结如下：

1. 使用下面的迭代函数求值找到最优上行功率分配 $\lambda_{i,j}$：

$$\lambda_{i,j} = \frac{1}{\left(1+\frac{1}{\gamma_{i,j}}\right)\mathbf{h}_{\mathbf{i,i,j}}^H\Sigma_{\mathbf{i}}^{-1}\mathbf{h}_{\mathbf{i,i,j}}} \tag{A-30}$$

其中

$$\Sigma_{\mathbf{i}} = \alpha_i\mathbf{I} + \sum_{m,n}\lambda_{m,n}\mathbf{h}_{\mathbf{i,m,n}}\mathbf{h}_{\mathbf{i,m,n}}^H \tag{A-31}$$

2. 根据最优上行功率分配 $\lambda_{i,j}$ 找到最优上行接收波束成形：

$$\hat{\mathbf{w}}_{\mathbf{i,j}} = \left(\sum_{m,l}\lambda_{m,l}\sigma^2\mathbf{h}_{\mathbf{i,m,l}}\mathbf{h}_{\mathbf{i,m,l}}^H + \sigma^2\alpha_i\mathbf{I}\right)^{-1}\mathbf{h}_{\mathbf{i,i,j}} \tag{A-32}$$

3. 通过缩放 $\hat{\mathbf{w}}_{\mathbf{i,j}}$ 找到最优的发射下行链路波束向量：

$$\mathbf{w}_{\mathbf{i,j}} = \sqrt{\delta_{i,j}}\hat{\mathbf{w}}_{\mathbf{i,j}} \tag{A-33}$$

该算法的全局收敛性由前面一节中讨论的对偶形结果和迭代函数求值的收敛性保证，后者可以通过类似 [5] 中那样的几行推理来证明。证明过程基于标准函数的性质 [38]。特别地，我们可以把对偶变量 $\lambda_{i,j}$ 装进一个向量 Υ，(30) 可以重写为

$$\lambda_{i,j}^{(t+1)} = f_{i,j}(\Upsilon^{(t)}),\ i = 1 \cdots N, j = 1 \cdots K \tag{A-34}$$

函数 $f_{i,j}$ 满足下面的性质：

1. 如果 $\lambda_{i,j} \geq 0 \forall i, j$，那么 $f_{i,j}(\Upsilon) > 0$。
2. 如果 $\lambda_{i,j} \geq \lambda'_{i,j} \forall i, j$，那么 $f_{i,j}(\Upsilon) > f'_{i,j}(\Upsilon)$。
3. 对 $\rho > 1$，我们有 $\rho f_{i,j}(\Upsilon) > f'_{i,j}(\rho\Upsilon)\ \forall i, j$。

它们在附录中会有论证。这些性质就保证了 $f_{i,j}$ 是一个标准函数。因此，从某个初值 $\Upsilon^{(0)}$ 开始，迭代函数求值算法会收敛到唯一的不动点，根据对偶性它一定是最优的下行功率。

A.4.2 最小化最大天线功率

为了解决最小化最大天线功率问题 (19)，我们使用定理 2 来解决对偶的具有不确定噪声的上行链路波束成形问题 (20)。不像总的带权发射功率最小化问题，求解最小化最大天线功率问题的对偶问题需要找到不确定的噪声的协方差阵 $\mathbf{Q_i}$ 和发射上行链路功率 $\lambda_{i,j}$。这里的想法是迭代地计算，内层最小化 $(\lambda_{i,j}, \hat{\mathbf{w}}_{\mathbf{i,j}})$，外层最大化 $\mathbf{Q_i}$。

内层最小化可以使用前一节展示的迭代函数求值的方法求解。至于外层最大化，我们使用了与 [23] 中呈现的类似的次梯度投影方法。考虑函数 $\phi(\mathbf{Q_1}, \cdots, \mathbf{Q_N})$，它是 (20) 固定 $\mathbf{Q_i}$ 的子问题：

$$\begin{aligned} \phi(\mathbf{Q_1}, \cdots, \mathbf{Q_N}) = \quad & \underset{\lambda_{i,j}}{\text{minimize}} \quad \sum_{i,j} \lambda_{i,j}\sigma^2 \\ & \text{subject to} \quad \Lambda_{i,j} \geq \gamma_{i,j} \forall i, j \end{aligned} \tag{A-35}$$

其中

$$\Lambda_{i,j} = \max_{\hat{\mathbf{w}}_{\mathbf{i,j}}} \frac{\lambda_{i,j} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,i,j}}|^2}{\sum_{(m,l)\neq(i,j)} \lambda_{m,l} |\hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{h}_{\mathbf{i,m,l}}|^2 + \hat{\mathbf{w}}_{\mathbf{i,j}}^H \mathbf{Q_i} \hat{\mathbf{w}}_{\mathbf{i,j}}}$$

[23] 表明了 ϕ 在 $(\mathbf{Q_1}, \cdots, \mathbf{Q_N})$ 中是凹的。更进一步，令 $\mathbf{w_{i,j}}$ 为最优的下行链路波

束向量，则 $\text{diag}\left[\sum_j \hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}}\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}}^H\right]$ 是 ϕ 对 $\mathbf{Q_i}$ 的次梯度。因此，外层最大化可以基于次梯度投影方法完成，其中投影投向约束集 $S_{\mathbf{Q_i}} = \{\mathbf{Q_i} : \text{tr}(\mathbf{Q_i}) \leq N_t, \mathbf{Q_i} \succeq 0\}$。

算法总结如下：

1. 对于 $i = 1, 2, \cdots, N$ 初始化 $\mathbf{Q_i}^{(0)}$
2. 固定 $\mathbf{Q_i}^{(n)}$。使用迭代函数求值找到最优的上行链路功率分配 $\lambda_{i,j}$：

$$\lambda_{i,j} = \frac{1}{\left(1 + \frac{1}{\gamma_{i,j}}\right)\mathbf{h}_{\mathbf{i},\mathbf{i},\mathbf{j}}^H [\Theta_{\mathbf{i}}^{(n)}]^{-1}\mathbf{h}_{\mathbf{i},\mathbf{i},\mathbf{j}}} \tag{A-36}$$

其中

$$\Theta_{\mathbf{i}}^{(n)} \triangleq \mathbf{Q_i}^{(n)} + \sum_{m,n} \lambda_{m,n}\mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{n}}\mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{n}}^H \tag{A-37}$$

3. 根据最优上行功率分配 $\lambda_{i,j}$ 找到最优上行接收波束成形：

$$\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}} = \left(\sum_{m,l} \lambda_{m,l}\sigma^2\mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{l}}\mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{l}}^H + \sigma^2\mathbf{Q_i}^{(n)}\right)^{-1}\mathbf{h}_{\mathbf{i},\mathbf{i},\mathbf{j}}$$

4. 通过缩放 $\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}}$ 找到最优的发射下行链路波束向量：

$$\mathbf{w}_{\mathbf{i},\mathbf{j}} = \sqrt{\delta_{i,j}}\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}} \tag{A-38}$$

5. 使用次梯度投影方法用步长 t_n 更新 $\mathbf{Q_i}^{(n)}$：

$$\mathbf{Q_i}^{(n+1)} = \mathscr{P}_{S_{\mathbf{Q_i}}}\left\{\mathbf{Q_i}^{(n)} + t_n\,\text{diag}\left[\sum_j \mathbf{w}_{\mathbf{i},\mathbf{j}}\mathbf{w}_{\mathbf{i},\mathbf{j}}^H\right]\right\} \tag{A-39}$$

投影操作是一个简单的重新归一化，即乘以一个常数使得 $\text{tr}(\mathbf{Q_i}^{(n+1)}) = N_t$。

6. 递增 n。回到第 2 步，直到收敛。

这个算法的全局收敛性由之前讨论到对偶性结果，迭代函数求值的收敛性，和次梯度投影方法的收敛性保证，后者是由于 $\phi(\mathbf{Q_1}, \cdots, \mathbf{Q_N})$ 是凹的 [39]。注意到当使用脏纸编码时，算法可以被简单地扩展过去。

A.4.3 分布式实现

第 IV-A 和 IV-B 小节中提出的迭代函数求值算法的一个重要特性是它们可以在 TDD 系统中用分布式的方式实现。TDD 系统中，上下行链路信道彼此互惠。在这种情形下，虚拟的对偶上行链路就是真实的上行链路。

首先考虑算法的迭代函数求值步骤。对上行功率 $\lambda_{i,j}$ 的函数迭代 (30) 包含有仅在各自小区内的基站通常知晓的信道向量 $\mathbf{h_{i,i,j}}$ 以及矩阵 $\mathbf{\Sigma_i}$。观察到 $\mathbf{\Sigma_i}$ 本质上是基站 i 处上行方向的接收信号的协方差阵，该信号包含要传送的信号、干扰和压缩后的背景噪声。在 TDD 系统中，这个协方差阵可以在上行方向上在每个基站本地进行估计。频道互惠意味着在上行接收机处的信号和干扰协方差矩阵恰好是所需的。此外，背景噪音水平通常是知道的，所以缩放因子 α_i 可以很容易补偿。因此，我们可以在每个小区一级对 $\lambda_{i,j}$ 和 $\mathbf{\Sigma_i}$ 估计值进行更新，也就是说，迭代函数求值过程可以在本地完成，无需明显的基站间协作。这个上行链路每个小区的迭代过程总是收敛的。它会收敛到最优的上行功率。注意到基站协作是通过上行功率控制（即所有 $\lambda_{i,j}$ 的更新，它会影响到所有其他的 $\mathbf{\Sigma_i}$）隐式地实现的。只有每个小区内的用户需要信道边信息，小区之外的干扰者并不需要。

其次，波束向量也可以很容易地在对偶的上行链路获得。这本质上就是基站处接收者的 MMSE 波束成形。

最后，要使用上行链路波束成形进行下行链路传输，我们需要把它以对的 $\delta_{i,j}$ 放缩。这包含一个矩阵求逆 (18)。不过，这个过程等价于一个下行链路中在有效的矩阵信道上为达到渴望的 SINR 集的功率控制问题。凭借 Foschini 和 Miljanic 的经典结果 [40]，通过基于每个用户的功率更新算法这个问题有一个分布式的实现。这个算法的每一步都设置一个 $\delta_{i,j}$ 以满足它相应的 SINR 约束取等号同时假定所有其他的 $\delta_{i,j}$ 都固定不变。该算法的收敛性可以通过 [40] 的方法或者标准函数参数 [38] 来证明。

事实上，算法中上行链路中每个小区的迭代函数求值和下行链路功率控制部分甚至可以在基站处使用可能过时的功率信息异步地实现。通过 [38] 中定理 4 表明的标准函数参数，这种异步更新的收敛性仍旧可以保证。唯一必要的同步是基站必须全部处于上行链路或者下行链路，以使得算法的三个步骤可以顺次执行。

上面的评注对于最小化最大天线功率问题同样适用。所需修改只是对 $\mathbf{Q_i}$ 的投影运算，即 (39)，和基于 $\mathbf{Q_i}$ 的接收协方差阵 (37)。二者都可以在一个小区的基础上完成。

A.4.4 波束成形 -功率迭代算法

解决下行链路波束成形问题的一个替代方法是 [19], [36] 中提出的迭代地更新波束成形和功率。最小化总的带权发射功率的算法如下：

1. 初始化 $\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}}$；
2. 找到满足 SINR 约束 (6) 取等号的 $\lambda_{i,j}$；
3. 根据上行功率分配 $\lambda_{i,j}$ 找到上行链路接收波束向量：

$$\hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}} = \left(\sum_{m,l} \lambda_{m,l} \sigma^2 \mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{l}} \mathbf{h}_{\mathbf{i},\mathbf{m},\mathbf{l}}^H + \sigma^2 \alpha_i \mathbf{I} \right)^{-1} \mathbf{h}_{\mathbf{i},\mathbf{i},\mathbf{j}} \tag{A-40}$$

4. 回到步骤 2 直到收敛；
5. 更新发射下行链路波束向量

$$\mathbf{w}_{\mathbf{i},\mathbf{j}} = \sqrt{\delta_{i,j}} \hat{\mathbf{w}}_{\mathbf{i},\mathbf{j}} \tag{A-41}$$

步骤 2 和 3 的迭代的收敛性在 [36] 中有证明。依据我们的多小区对偶性结果，这个算法一定也是收敛到下行链路的全局最优解。

迭代函数求值算法和波束成形 -功率迭代算法都可以给出多小区下行链路波束成形问题的最优解。二者都可以直接在 TDD 系统中实现。然而波束成形 -功率迭代算法的实现需要在上面第 2 步对上行链路功率重复更新，这本身需要集中式的处理（例如通过进行矩阵求逆）或者单独的迭代过程（例如通过不动点迭代）。由于上面的步骤 2 和步骤 3 都需要所有用户重复的在同一时间执行，波束成形 -功率迭代算法相比于前一节提出的迭代函数求值算法来说需要所有用户所有基站之间更高水平的同步。

A.5 仿真

我们以研究图 2 所示的两小区设置下的协作波束成形的好处开始。仿真中使用的是标准的蜂窝网络参数：噪声功率频谱密度设为 $-162\,\mathrm{dBm/Hz}$；信道向量的选取依据距离相关的路径损耗 $L = 128.1 + 37.6\log_{10}(d)$，其中距离 d 单位是千米，有 8 dB 对数正态的遮蔽效应，和瑞利项。相邻基站的距离设为 2.8 km。天线增益定为 15 dBi。为说明起见，基站天线功率约束的权重设为 $\alpha_i = 1$ 。

图 3 显示了每个小区内有两个用户时协作波束成形的收益。图 4 显示了类似的情形，不过每个小区内有三个用户时。两种情形下，每个小区内的有一个

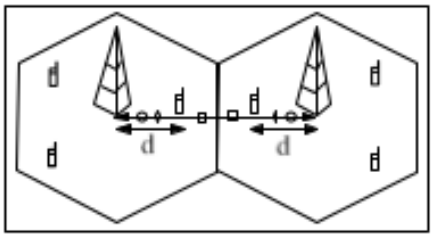

图 A.2　两小区设置，两个用户位于两个基站间，距离为 d。

用户位于两个基站相连直线上，到自己基站的距离为 d。其他用户随机地分布在小区内的其他地方。基站处配有四个天线。从图中可以明显地看出协作波束成形系统比起传统的单小区优化系统性能明显提升，尤其是在高目标 SINR 范围内。如之前所料，当用户紧邻小区边缘时收益最大。

图 3 和图 4 还显示了协作波束成形带来的性能增益在每个小区两个用户的情形下比在每个小区三个用户的情形下要大。直观上讲，当每个小区的用户数目相对于基站天线数目较小时，会有空余的维度可以用来做干扰抑制。这时协作波束成形的收益最为显著。

注意到一个有趣的现象：在每个小区两个用户的情形下，随着目标 SINR 增大，传统系统最终会变得不可行。然而协作波束成形系统总是可行的。这是因为基站配有四个天线。当两个小区总共有四个用户时，协作系统有能力把小区外用户迫零，从而完全消除小区外干扰。相反地，在传统系统中小区外干扰总是存在的。注意到在每个小区三个用户的设定下，完全的迫零不再可能。但是协作波束成形仍然可以带来显著的功率节省。

图 5 和图 6 显示了在一个七小区每个小区三个随机分布的用户的场景下（如图 1）在最小化总发射功率以及最小化最大天线功率准则下协作波束成形的性能。每个基站仍然配有四个天线。我们可以再次观察到在高目标 SINR 范围内联合最优化算法比传统单小区更新性能要好。这时因为在高 SINR 情况下，多小区网络主要是干扰受限的。两图还显示了脏纸编码对联合最优化和单小区更新算法二者带来的增益。

图 7 显示了使用逐天线最优化算法带来的最大天线功率的节省。对于这个七小区每个小区三个用户的情形，功率节省在 1–2 dB 范围内。

为了显示我们提出的迭代函数求值算法的收敛性，我们将它和 [19] 中的波束成形 -功率迭代算法进行了比较。图 8 显示了上行链路发射功率（单位mW）

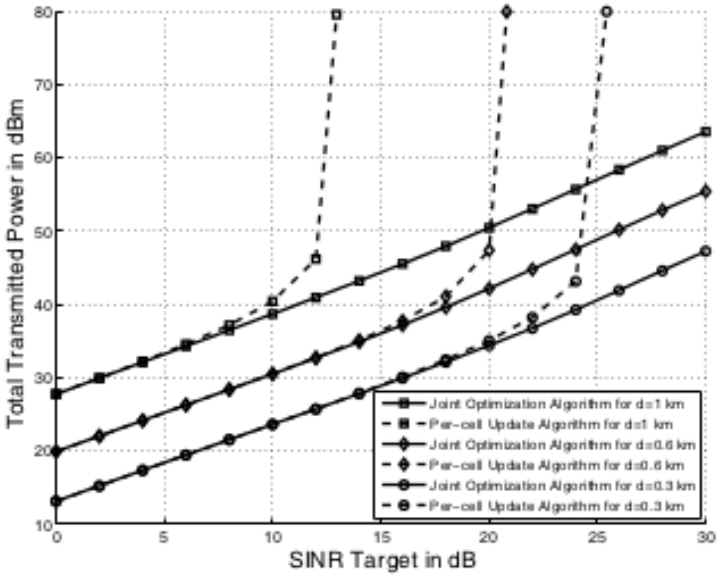

图 A.3　两小区网络每个小区两个用户情形下使用协作波束成形系统联合最优化和传统系统单小区更新时总发射功率与目标 SINR 的关系。用户之一距离自己的基站 d 取了若干个值。基站配有四个天线。

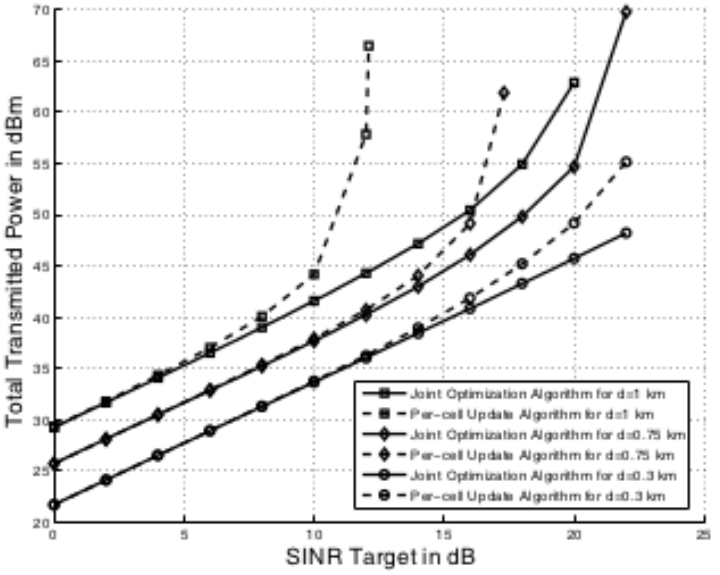

图 A.4　两小区网络每个小区三个用户情形下使用协作波束成形系统联合最优化和传统系统单小区更新时总发射功率与目标 SINR 的关系。用户之一距离自己的基站 d 取了若干个值。基站配有四个天线。

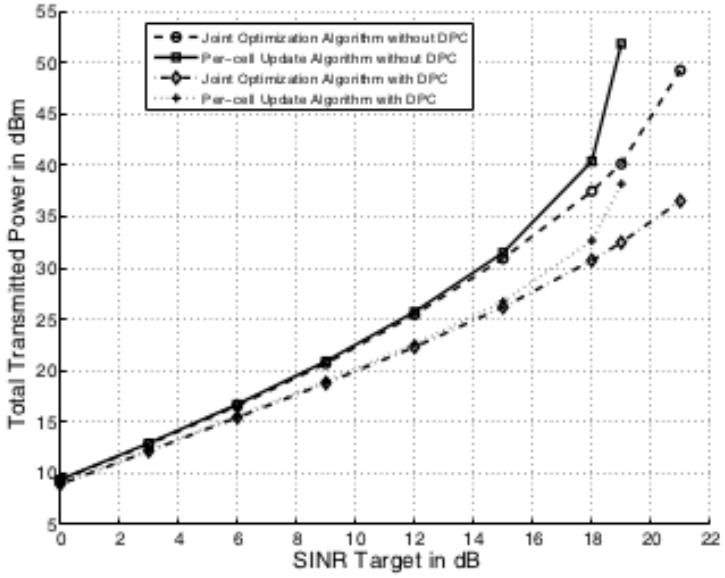

图 A.5　七小区网络每个小区三个用户情形下使用协作波束成形系统联合最优化和传统系统单小区更新时使用及不使用脏纸编码时总发射功率与目标 SINR 的关系。基站配有四个天线。

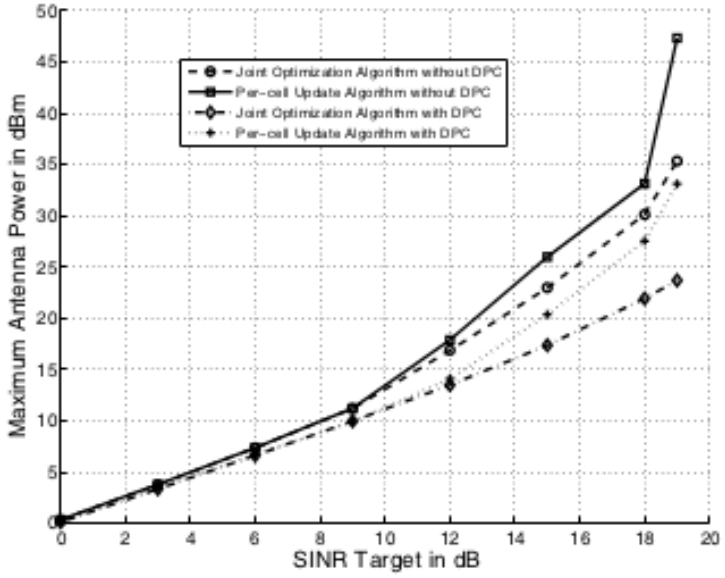

图 A.6　七小区网络每个小区三个用户情形下使用协作波束成形系统联合最优化和传统系统单小区更新时使用及不使用脏纸编码时最大单天线功率与目标 SINR 的关系。基站配有四个天线。

的范数残留与迭代次数的关系。范数残留定义为：

$$R^{(n)} = \sigma^2 ||\Upsilon^{(n)} - \Upsilon^*||_2 \tag{A-42}$$

其中 Υ^* 代表了最优的功率向量。可以观察到尽管波束成形 -功率迭代算法在最初收敛地更快，分布式的迭代函数求值算法事实上更快地渐进收敛。注意对于分布式的迭代函数求值算法，每个迭代步骤都需要估计基站处接收信号的协方差阵。所以算法的收敛速度往往会受估计过程的准确性影响，尤其是当信道随时间改变时。

最后，我们注意到迭代函数求值算法的收敛速度不仅依赖于问题的规模（例如天线数、小区数和每个小区的用户数），而且依赖于目标 SINR。图 9 显示了对于不同的目标 SINR 值范数残留与迭代次数的关系。可以观察到，当目标 SINR 增大时收敛会变慢。

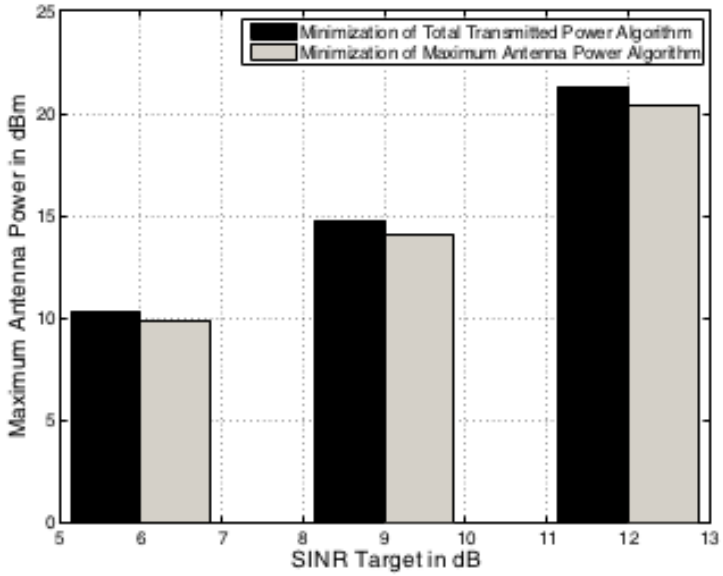

图 A.7　总功率最小化算法和最大天线功率最小化算法的最大天线功率比较

A.6　结论

本文为多小区每个小区多用户的网络中的最优协作下行链路波束成形设计问题提供了一个解决方案。使用拉格朗日理论上下行链路对偶性被推广到了多小区情形下的两个不同的设计准则：SINR 约束下最小化总的带权发射功率和最

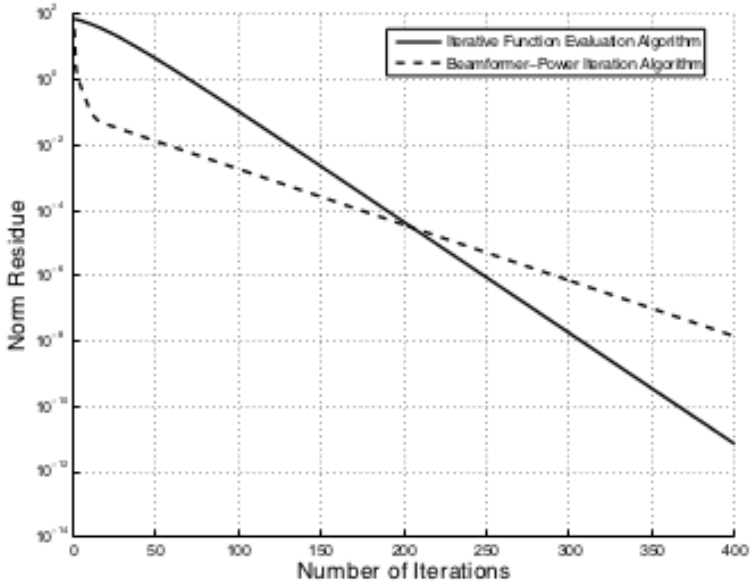

图 A.8　两种最优化算法下范数残留对迭代次数的关系

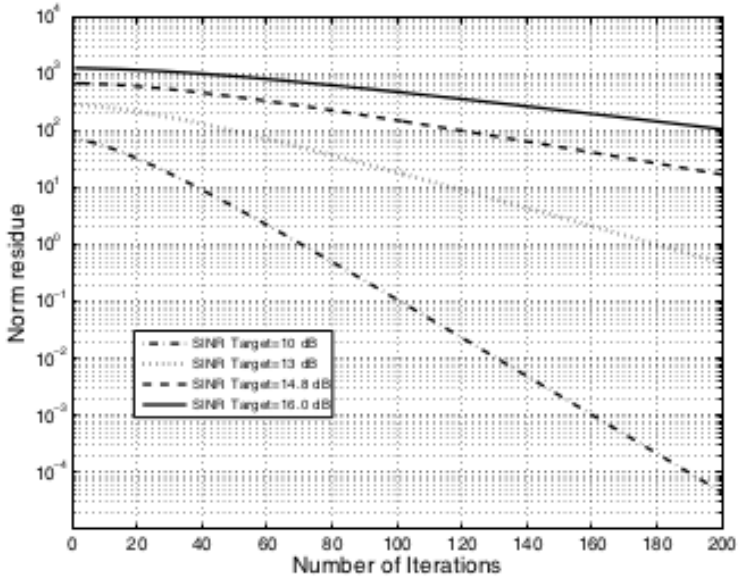

图 A.9　不同目标 SINR 下范数残留与迭代次数的关系

小化最大天线功率。我们提出了一个基于迭代函数求值的算法，它能够找到全局最优的协作波束成形。算法的关键特性是它可以在 TDD 系统中以分布式的方式实现。这个算法是高效的，它的性能优于传统的单小区信号处理的无线系统。

书面翻译对应的原文索引

[1] Dahrouj H, Yu W. Coordinated beamforming for the multicell multi-antenna wireless system. IEEE Transactions on Wireless Communications, 2010, 9(5):1748–1759