西北工業大學

本科毕业设计论文

论文（设计）题目：

基于仿射投影算法的信道评估技术研究

专业名称 信息对抗技术

学生姓名 肖 雅 楠

指导教师 智 永 锋

毕业时间 2013 年 7 月

原 创 性 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，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名：_____ 日 期：_____

关于学位论文使用授权的声明

本人完全了解西北工业大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权西北工业大学可以将本学位论文全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

（保密的论文在解密后应遵守此规定）

论文作者签名：_____ 导师签名：_____ 日 期：_____

摘 要

计算机和数字通信已经对当今社会生活的各个方面产生了前所未有的影响。为了满足人们对数字通信服务日益增长的需求，工程师必须采取便捷可行、经济适用的系统分析设计方法。在数字通信领域，从传输内容到传输媒介以及各种辅助软件，我们能看到这些可用技术数量上的大规模增长。但是在炙手可热的系统性能设计、分析和优化领域，经典好用的技术并不多，实际工程效果也不尽如人意。在过去的几十年里，形形色色的计算机辅助工程技术被开发出来用于设计复杂系统。这些计算机辅助工程技术十分依赖于对复杂系统各个部件及系统本身的建模、分析和仿真—这是持续整个项目生命周期的过程。在电子计算机走进千家万户的今天，对通信系统领域的计算机辅助设计、分析和模拟进行更加深入和细致的研究，实不为过。

本文通过对数字通信系统领域一些基本概念的探讨、基本方法的回顾和几个基础模拟方法的比较，研究了该领域的几个重要方向。本文的核心部分是蒙特 –卡罗方法、仿射投影算法和其他一些方法在通信系统的中间环节 —信道部分的分析。并且，为了更好地将比较结果展示出来，本文列举出了几种图形化的分析结果 —从本文作者毕业设计的研究经历来看，在通信系统领域图形化是对研究成果一种极好的表示。

关键词: 仿射投影算法，性能分析，信道评估，通信，MATLAB

ABSTRACT

Digital communications and computers are having a tremendous impact on the world today. In order to meet the increasing demand for digital communication services, engineers must design systems in a timely and cost-effective manner. The number of technologies available for providing a given service is growing daily, covering transmission media, devices, and software. The resulting design, analysis and optimization of performance can be very demanding and difficult. Over the past decades, a large body of computer-aided engineering techniques have been developed to facilitate the design process of complex technological systems. These techniques rely on models of devices and systems, both analytic and simulation, to guide the analysis and design throughout the life cycle of a system. Computer-aided design, analysis and simulation of communication systems constitute a new and important part of this process.

This thesis studies different aspects of the simulation of communication systems by covering some basic ideas, approaches and methodologies within the simulation context. Performance measurement of a digital communication is the main focus of this thesis. However, some popular visual indicators of signal quality, which are often generated in a simulation to provide a qualitative sense of the performance of a digital system, are also considered.

Another purpose of this thesis is to serve as a model for developing simulation or template of other systems. In other words, peer researchers learning to simulate a system can use the work presented here as a starting point.

Key Words: affine projection algorithm, performance evaluation, channel estimation, communication, MATLAB

目 录

第一章　绪论

近年来，随着超大规模集成电路技术的飞速发展以及人们对更加高速、稳定的数字通信系统需求的不断增长，通信系统和信号处理系统的复杂度也在日益增加。这种程度的发展让传统的系统设计方法显得捉襟见肘，更不用说动态地分析系统设计中出现的各种问题。为此，研究人员在不断尝试新的、高效的设计分析工具 —在电子计算机日益普及的时代，计算机辅助的系统模拟方法 (Simulation) 正逐渐从实验室走向市场，被企业界认可。更高级的系统仿真方法 (Emulation) 限于作者的研究时间和精力，将在工作展望部分进行适当阐述。

通信系统的模拟，主要是通过数字计算机辅以相应的分析软件，对系统特性的一个或多个方面进行“模拟”，也就是计算。通信系统模拟的研究人员的设想是，假如物理的（现实生活中的）通信系统的每一个要素都可以用一个数学模型来表示，那么数字计算机将会“变身”成为一个完全虚拟的通信环境 —只需要按照现实情况把各个要素连接起来。如摘要中所述，这样纯粹的虚拟环境越来越受研究人员欢迎，很重要的一点是模型中的每一个参数可以根据需要随时进行调整，调整后的结果也能很方便的观测到 —这大大简化了用传统的方法逐步逼近最优参数的工作量。

1.1　研究背景

最早的控制系统模拟可以追溯到 20 世纪 50 年代初期，电子计算机刚刚诞生的时候。由于使用的仍是模拟器件，当时的计算机可被视为对一个连续系统的模拟. 该连续系统由大量互相接在插线板上的模块构成，而这些模块，分别代表了系统框图[1]上的不同部分。框图上的线性元件，诸如积分器、信号求和点以及放大器均通过功率放大器来进行模拟。非线性元件，诸如限制器和一些非线性函数，则可以通过几个线性、非线性元件的组合来表示。由以上推理可知，在数字通信领域，任何一个系统，无论它的特性是由常数或时变 (time varying) 参数的线性、非线性微分方程来表示，都能够被化简成一个框图 (block diagram)，进而通过计算机软件提供的组件搭接模拟。模拟的系统搭接好之后，根据给定输入信号（一称激励信号，exciting signal）的不同，研究人员可以在输出端观测到系统的各种动态特性[2]。

从 20 世纪 60 年代开始，技术的进步让原本纯粹模拟 (analog) 的系统特性分析方法渐渐向数字化 (digital) 转移。数字化的系统模拟架构由诸如连续系统建模程序 (Continuous System Modeling Program, CSMP) 和其他一些面向模块的编程语言搭建。这些编程语言通过一个部件、一个部件的分析方式来完成对模拟计算机的仿真。研究人员开发出这些编程语言的最初动力是将繁杂的、模拟的分析方式升级成简单、方便的分析方式 —用几行代码，就可以将一堆框图所要传递的信息表示出来，结果

还是一样的。上世纪 60 年代中叶，成套的电路级模拟软件诸如 SPICE 和 ECAP 被开发出来，这些软件大大降低了系统分析中数值积分和信号流图中拓扑化简的工作量。从加州大学伯克利分校电子工程和计算机科学系的主页上可以看到，经历了半个世纪，SPICE 在教学中仍是极好的工具[3]。

从 20 世纪 60 年代末到 70 年代初，离散时间系统和数字信号处理领域的发展带来了又一波数字方法的系统分析浪潮。随之而来的分析软件有 SYSTID,CSMP,CHAMP,LINK 等等。这些软件中的部分在设计和分析卫星通信链路的时候发挥了巨大的作用[4]。

从 20 世纪 80 年代开始，随着软件工程理论的不断推进，带有菜单的交互式系统分析软件如 ICSSM,TOPSIM 和 ICS 被开发出来。这些软件让仿真从命令行窗口进入了图形化时代。

自 20 世纪 80 年代中叶起，通信系统的模拟以及仿真在电子工程学科下作为一个独立的领域受到越来越多的关注。这些年计算机的软硬件仍取得了很大的发展，并且再可预见的几十年里还有很大的发展空间。硬件方面，个人计算机已经成为了一般研究人员的主流分析工具；软件方面，从 20 世纪 90 年代的 BOSS，SPW，COSSAP 等软件，到今天占据绝对多数的 MATLAB[5] （有几乎覆盖科研全领域的工具箱）提供交互式用户友好图形化模拟仿真环境。

1.2 章节安排

本文主要研究了数字通信系统的建模和计算机模拟。引入的几个主要研究方法是，蒙特卡洛方法 (Monte Carlo method)、准分析方法 (Quasi Analytical method)、仿射投影算法[6](Affine Projection Algorithm, APA)。本文对这几种方法在信道评估领域做出了极为细致的比较。除此之外，本文也适当采用了其他一些方法，会随着行文逐渐展开，不一一赘述。

作为一篇本科生毕业设计的学位论文，作者觉得还是很有必要阐述一些通信，尤其是数字通信领域的基础知识。一是为了所有可能参考这篇文献的人行方便，二是为了巩固作者自身的学科基础 —工程和科学领域什么东西都能和数学沾点边，而牢固的数学基础又是做出优秀成果的有力保证。

第二章介绍了数字通信系统模拟的背景知识。诸如常见信号的离散表示以及带通信号的低通表示。作为本文研究的主要内容，信道的基本概念和几种常见的信道模型也做了一定的介绍。

第三章讨论了仿射投影算法在自适应滤波 (Adaptive Filtering) 中的应用。从该算法的基础知识，到应用在自适应滤波上的一些最基本的性能分析都有所讨论，

第四章和第五章是本文的核心部分。第四章分析了整个信道评估方法设计的全过程，从信道建模到最后结果的评估方案。并且构造了几种有效的机制对几种分析

方法做了横向比较。

第五章是结果阐述部分。对前几章提出的各种分析进行了 MATLAB 模拟计算，并依据行业惯例，绘制了若干张更加直观的评价系统特性的图。

第六章对全文所有工作进行了一个总结。然后对论文最终稿的成果进行了分析，列举出了几个可以从本文继续延伸拓展研究的方向。

第二章　背景知识

通常来说，系统可以被视为为了完成特定任务组合联结而成的几个部件[7]。信号则可被视为，它通常是在时间维度上的一些量的集合。在通信领域用的信号一般是电压或者电流，本文实验中用的信号是随机信号，由于不涉及实际意义，仅作分析方法是用，故不带单位。

我们所期望模拟的现实生活中的信号，占绝大多数的是连续信号。但是，受限于数字计算机的物理特性，在进行模拟的时候只有将模拟信号进行采样 (sampling) 和量化 (quantizing) 处理。

2.1　连续和离散信号

连续信号，顾名思义，就是在时间上是连续的信号。它的数学定义为一个以连续时间为自变量的实数或复数域函数。在连续信号里，人们定义了几个十分基础的信号，例如单位阶跃信号 (unit step function)、单位脉冲信号 (unit pulse signal) 和 sinc 信号。这些信号从定义到推导到理论应用，在每一本《信号与系统》教材上都有所阐述，因此本文不再赘述。

离散信号与连续信号不同的地方在于，离散信号仅仅定义在一个或多个离散的时间点上。最常见的一种离散信号获得方式是对连续信号进行等间隔采样 $t = nT_s$，其中 n 为整数。

我们希望从数学上可以根据采样到的信号恢复出原信号（即采样前的信号，连续信号）。采样定理（一称奈奎斯特定理）从理论上指出了最小的采样频率。根据采样定理，任意带宽为 W 的连续信号 $s(t)$ 能够在采样后恢复成原信号的最小采样频率为 $f_s \geq 2W$ 采样每秒。满足条件的最小采样频率 $f_n = 2W$ 采样每秒被称为奈奎斯特频率。因此，从理论上可以看出，现实生活中的任何一个信号都可被以不低于奈奎斯特频率的采样频率采样，然后量化成离散信号。这样得到的离散信号，在数字通信系统分析时，并不会比直接用原信号有任何失真，还降低了计算复杂度。

2.2　带通信号的表示

根据参考文献[2]，数字通信系统的两大基本理论分别为采样定理（见2.1节）和欧拉定理。欧拉定理将实数域和复数域通过公式 $e^{ix} = \cos(x) + i\sin(x)$ 联接起来。式中 e 为自然对数的底，i 为虚数单位。作为一篇本科生学位论文，本文仍会用一定篇幅来阐述与之相关的概念和基础应用，为后文设计和验证实验做铺垫。

从教材和现实生活经验中我们知道，信号（相关讨论在2.1节）可以被视为承载

着特定信息的数学量。在实际应用中，数字信息承载信号 (digital information-bearing signal，以下简称信息信号) 通常需要经过某种类型载波 (carrier) 的调制然后才进行传输。当信息信号的带宽远远小于载波频率时，信息信号就可以被视为窄带带通信号 (narrowband bandpass signal，简称窄带信号)。我们将对后文需要用到的公式做一个推导，力求获得最简形式。

如前所述，当采样频率大于或等于奈奎斯特频率的时候，任意连续信号都可以用其等价的离散信号予以表示。这也就是说，假设某带通连续信号的带宽为 B，其载波频率为 f_c，那么不失真的最小采样频率为 $2\times(f_c+B/2)$，依据为前文2.1推导的采样定理。当然，如果仅仅讨论带宽为 W 的低通信号，其奈奎斯特频率便只是 $2W$。由参考文献[8] 可知，对载波调制的信号和相应的系统做一定限制，它们就能被分别当做低通的对象来进行处理。下面进行适度的公式推导，假设采用的为最常见的余弦调制。

任意余弦调制后的信号 $x(t)$ 可以表示为

$$x(t) \;=\; r(t)\cos[2\pi f_c t+\varphi(t)] \tag{2–1}$$

上式简化后可以表示为

$$x(t) \;=\; Re[r(t)e^{j\phi(t)}e^{j2\pi f_c t}] \tag{2–2}$$

其中 j 为虚数单位，$r(t)$ 为幅值调制 (amplitude modulation)，$\varphi(t)$ 为信号的相位调制 (phase modulation)，f_c 为调制用的余弦载波的频率。那么，分析后不难发现信号

$$v(t) \;=\; r(t)e^{j\varphi(t)} \tag{2–3}$$

包含了信息信号 $r(t)$ 的所有信息，并且经过傅里叶变换[9] (Fourier Transform) 后可以看出其低通特性。上述方法通常被称为复数域低通特性或者是信号的复数包络检测。如果任意信号满足本章前述的窄带特性，那么将调制后的带通信号 $x(t)$ 处理为低通信号 $v(t)$，然后作为信道分析的激励信号，会在计算上有很大的方便。

我们利用三角函数的和差化积与积化和差公式[9]，对上面得到的公式进行进一步的分析，可以得到

$$x(t) \;=\; x_d\cos(2\pi f_c t)-x_q\sin(2\pi f_c t) \tag{2–4}$$

其中 x_d 与 x_q 分别为带通信号 $x(t)$ 在正交的而为坐标轴上的两个分量 (In-Phase, Quadrature)。从参考文献 [8] 中可以得知，如果该带通信号 $x(t)$ 的带宽 B, 也就是其两个分量的带宽 B 满足条件 $B\ll f_c$，那么 $x(t)$ 的希尔伯特变化可以视为

$$\tilde{x}(t) \;=\; x_d\sin(2\pi f_c t)+x_q\cos(2\pi f_c t) \tag{2–5}$$

进而信息信号 $x(t)$ 的分析式可以表示为

$$x_+(t) \;=\; x(t)+j\tilde{x}(t)=[x_d(t)+jx_q(t)]e^{j2\pi f_c t} \tag{2–6}$$

进而载波调制的信号 $x(t)$ 可以表示为

$$x(t) \;=\; Re[x_+(t)] = Re[\tilde{x}(t)e^{j2\pi f_c t}] \tag{2–7}$$

综合以上各式，信息信号 $x(t)$ 的等价低通表示，或者说复数域包络检测定义为

$$\tilde{x}(t) \;=\; x_+ e^{-j2\pi f_c t} = x_d(t) + jx_q(t) \tag{2–8}$$

2.3 M 维相移键控调制信号

根据参考文献[8, 9] 中提供的信号分析方法，任意一个 M 维相移键控调制信号 (M-*ary* Phase-Shift Keying Signal) 可以表示为

$$s(t) \;=\; A\cos[2\pi f_c t + \frac{2\pi}{M}(m-1) + \theta_i] \tag{2–9}$$

其中 $1 \le m \le M$，$0 \le t \le T$ 并且 θ_i 是信号在发送端 (transmitter) 加上的相角 (phase angle)。公式(2–9)可以简化表示为

$$s(t) \;=\; ARe\{e^{j[\frac{2\pi}{N}(m-1)+\theta_i]}e^{j2\pi f_c t}\} \tag{2–10}$$

由公式(2–7)及参考文献[8, 9] 可知，复数形式的包络函数（包络检测获得）为

$$\tilde{s}(t) \;=\; Ae^{j[\frac{2\pi}{M}(m-1)+\theta_i]} \tag{2–11}$$

因此该包络函数的正交分量分别为

$$x_d(t) \;=\; A\cos[\frac{2\pi}{M}(m-1) + \theta_i] \tag{2–12}$$

和

$$x_q(t) \;=\; A\sin[\frac{2\pi}{M}(m-1) + \theta_i] \tag{2–13}$$

2.4 M 维频移键控调制信号

根据参考文献[8, 9] 中提供的信号分析方法，任意一个 M 维频移键控调制信号 (M-*ary* Frequency Shift Keying Signal) 可以表示为

$$x(t) \;=\; A\cos[2\pi(f_c + I_m\Delta f)t + \theta_m] \tag{2–14}$$

或者其复数域表示

$$x(t) \;=\; ARe\{e^{j2\pi f_m \Delta f}e^{j\theta_m}e^{j2\pi f_c t}\} \tag{2–15}$$

其中 $1 \le m \le M$，$\Delta f = \frac{1}{T}$。并且 Δf 是为了保证信号正交性的最小频率间隔。此外，还有 $I_m = \frac{2m-1-M}{2}$。θ_m 表示的是每一个信号集中，信号的相位。

由公式(2–7)及参考文献[8, 9] 可知，M 维频移键控调制信号的包络函数为

$$\tilde{x}(t) = Ae^{j(2\pi f_m \Delta f t + \theta_m)} \tag{2–16}$$

可得其正交分量为

$$x_d(t) = A\cos(2\pi I_m \Delta f t + \theta_m) \tag{2–17}$$

及

$$x_q(t) = A\sin(2\pi I_m \Delta f t + \theta_m) \tag{2–18}$$

2.5 通信系统中的滤波

通信系统中的滤波主要用来选择特定的信号以及减轻或消除干扰 (interference) 和噪声 (noise)。一个理想的滤波器，在其通带（也就是进行滤波的频段）传递函数的幅值 (magnitude) 是恒定不变的一个值；在其阻带，传递函数的幅值为 0. 理想滤波器的中频带增益为 1，其相位为关于频率的线性函数[7]。

在对数字通信系统进行模拟的时候，带通信号的滤波通常都简化为对其复数域的包络函数进行处理。带通滤波器一般也等价成低通滤波器方便分析。前文1.1节中分析过，在 MATLAB 软件中，各式各样的滤波器都只需要几个参数，诸如通带 (passband)、阻带 (stopband) 和这些频带中的公差 (tolerances) 就可以确定下来。在 MATLAB 中最常见的描述方式为传递函数 —分子和分母都是多项式。

第三章　仿射投影自适应滤波器

自适应滤波器在统计信号处理中占有相当重要的地位。当需要处理统计特性未知（unknown statistics）或者根本就不固定环境中产生的信号时，自适应滤波器相比用传统方法设计的固定滤波器 (fixed filters) 在处理结果的精度和可靠性上有显著的改善，因此采用自适应而非传统滤波器在这些环境中是极富吸引力的。进一步来说，虽然从理论上来看自适应滤波器的处理速度没有传统的快，但是经过多年的发展，目前最先进的自适应滤波器已经完全不输传统滤波器，在处理速度方面；在系统的复杂度方面，自适应滤波器由于其结构特性，略逊一筹也是不争的事实。从自适应滤波器诞生的那一天起，越来越多的实际应用中可以看到它的身影。现如今，应用领域已包括诸如通信中的雷达和声呐，控制领域的各种新型控制器、地震的预测以及生物医学工程领域的医学图像分析、病情诊断和预测等。

本章主要讨论了仿射投影算法的数学基础和其在自适应滤波器中的应用[10]。滤波器的自适应特性 (adaptation) 是通过根据输入量（在通信系统中为输入信号）的变化来不断调整其自由参数 (free parameters) 也即其传递函数自变量的系数，来获得的。然而，这样一种方式，却使得自适应滤波器成了一种非线性器件。所以，当我们说自适应滤波器是一种线性器件的时候，我们是如下定义的：自适应滤波器的输入输出映射满足叠加定理，当且仅当在每一个瞬时，滤波器的各个参数均为定值的时候。

3.1　仿射投影自适应滤波器的建立

从数学角度上讲，设计一个基于仿射投影算法的自适应滤波器，就是一个在有多重限定条件的前提下试图寻找最优解的过程。从下文开始我们慢慢讨论。

试图将权重矢量增量的平方欧几里得范数最小化，我们可以得到下式

$$\delta\hat{\mathbf{w}} \;=\; \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) \tag{3–1}$$

其 N 维限定条件是

$$d(n-k) \;=\; \hat{\mathbf{w}}^H(n+1)\mathbf{u}(n-k) \quad for \quad k = 0, 1, ..., N-1 \tag{3–2}$$

对于式(3–2)我们要求，也可以说成假设 N 要小于输入数据空间的维度，即 M。输入空间在这里也就是权重空间 (weight space) 。

在这样一种 N 维的限定条件下，当 $N = 1$ 时，就变成了归一化最小均方滤波器 (Normalized Least Mean Square Filter, 简称 NLMS Filter)。因此，我们可以将限定条件的数量 N 视为，基于仿射投影算法的自适应滤波器的阶数 (order)。

根据参考文献[10] 的附录 C (pp 799-801) 中描述的多重约束条件下拉格朗日乘子的计算法则，我们将方程(3–1)和(3–2)组合起来建立仿射投影自适应滤波器的代价函数

$$\mathbf{J}(n) \;=\; \|\hat{\mathbf{w}}(n+1)-\hat{\mathbf{w}}(n)\|^2+\sum_{k=0}^{N-1} Re[\lambda_k^*(d(n-k)-\hat{\mathbf{w}}^H(n+1)\mathbf{u}(n-k))] \quad (3\text{–}3)$$

在代价函数(3–3)中，λ_k 是关于多重限定条件的拉格朗日乘子。为了在后面中的描述简便起见，我们接着引入如下定义，

- 一个 N 行 M 列的数据矩阵 (data matrix)，其埃尔米特转置 (Hermitian transpose) 定义为
$$\mathbf{A}^H \;=\; [\mathbf{u}(n),\mathbf{u}(n-1),...,\mathbf{u}(n-N+1)] \quad (3\text{–}4)$$
- 一个 N 行 1 列期望响应矢量 (desired response vector)，其埃尔米特转置定义为
$$\mathbf{d}^H(n) \;=\; [d(n),d(n-1),...,d(n-N+1)] \quad (3\text{–}5)$$
- 一个 N 行 1 列的拉格朗日矢量 (Lagrange vector)，其埃尔米特转置定义为
$$\boldsymbol{\lambda}^H \;=\; [\lambda_0,\lambda_1,...,\lambda_{N-1}] \quad (3\text{–}6)$$

将上述定义好的矩阵代入方程(3–3)中，一个简化后更加紧凑的代价函数为

$$\mathbf{J}(n) \;=\; \|\hat{\mathbf{w}}(n+1)-\hat{\mathbf{w}}(n)\|^2+Re[(\mathbf{d}(n)-\mathbf{A}(n)\hat{\mathbf{w}}(n+1))^H\boldsymbol{\lambda}] \quad (3\text{–}7)$$

为了求极值，我们用参考文献[10] 的附录 B (pp 794-798) 中给出的关于复数域矢量求微分的方法，对代价函数 $\mathbf{J}(n)$ 求关于权重矢量 $\hat{\mathbf{w}}(n+1)$ 的导数，有

$$\frac{\partial \mathbf{J}(n)}{\partial \hat{\mathbf{w}}^*(n+1)} \;=\; 2(\hat{\mathbf{w}}(n+1)-\hat{\mathbf{w}}(n))-\mathbf{A}^H(n)\boldsymbol{\lambda} \quad (3\text{–}8)$$

将上式右边置为 0，我们可以得到等式

$$\delta\hat{\mathbf{w}}(n+1) \;=\; \frac{1}{2}\mathbf{A}^H\boldsymbol{\lambda} \quad (3\text{–}9)$$

为了讨论和求解的需要，我们想办法从方程(3–9)中消去拉格朗日矢量 λ。因此，我们首先利用方程(3–4)和(3–5)将最开始的 N 维限定条件集重写为等价的形式，

$$\mathbf{d}(n) \;=\; \mathbf{A}(n)\hat{\mathbf{w}}(n+1) \quad (3\text{–}10)$$

将方程(3–9)两边同时乘以 $\mathbf{A}(n)$，然后利用方程(3–1) 和(3–10)消去更新后的权重矢量 $\hat{\mathbf{w}}(n+1)$ 得到

$$\mathbf{d}(n) \;=\; \mathbf{A}(n)\hat{\mathbf{w}}(n)+\frac{1}{2}\mathbf{A}(n)\mathbf{A}^H(n)\boldsymbol{\lambda} \quad (3\text{–}11)$$

从上式(3–11)我们可以推出如下结论：

- 矢量 $\mathbf{d}(n)$ 和 $\mathbf{A}(n)\hat{\mathbf{w}}(n)$ 的差，是一个以迭代次数 (iteration) 为自变量的 N 行 1 列误差矢量 (error vector)

$$\mathbf{e}(n) \;=\; \mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n) \tag{3–12}$$

- 矩阵 $\mathbf{A}(n)\mathbf{A}^H(n)$ 的乘积是一个 N 行 N 列的矩阵，我们将其逆矩阵记作 $(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}$

有了以上简化的表示，我们解方程(3–11)求拉格朗日矢量，

$$\boldsymbol{\lambda} \;=\; 2(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n) \tag{3–13}$$

将解得的拉格朗日矢量代入方程(3–9)，我们可以得到权重矢量的最优增量，其表示形式为，

$$\delta\hat{\mathbf{w}}(n+1) \;=\; \mathbf{A}^H(n)\mathbf{A}(n)\mathbf{A}^H(n)^{-1}\mathbf{e}(n) \tag{3–14}$$

最后，也可以说是在基于仿射投影算法的自适应滤波器设计中最重要的一步，就是我们需要控制好每一步权重矢量的迭代，使其尽可能保证在一个方向上[6] (keep the same direction)。为了尽可能的实现这一目标，我们向方程(3–14)中引入步长参数 $\tilde{\mu}$ (step-size parameter), 进而得到新的方程

$$\delta\hat{\mathbf{w}}(n+1) \;=\; \tilde{\mu}\mathbf{A}^H(n)\mathbf{A}(n)\mathbf{A}^H(n)^{-1}\mathbf{e}(n) \tag{3–15}$$

将获得的每一步迭代的权重矢量增量代入其定义式（也就是其每步迭代的递推公式）(3–1) ，并进行整理，可以得到

$$\hat{\mathbf{w}}(n+1) \;=\; \hat{\mathbf{w}}(n) + \tilde{\mu}\mathbf{A}^H(n)\mathbf{A}(n)\mathbf{A}^H(n)^{-1}\mathbf{e}(n) \tag{3–16}$$

这也就是最终我们利用基于仿射投影算法的自适应滤波器，在数字通信系统中进行信号处理的基本公式 [6, 10]。

3.2 仿射投影运算

由参考文献上的描述[10] 可得，迭代后的权重矢量 $\hat{\mathbf{w}}(n+1)$ 是迭代前权重矢量 $\hat{\mathbf{w}}(n)$ 经过仿射投影运算的结果。为了得到仿射投影算子，我们将方程(3–12) 代入方程(3–16)可得，

$$\begin{aligned}\hat{\mathbf{w}}(n+1) \;=\; &[\mathbf{I} - \tilde{\mu}\mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{A}(n)]\hat{\mathbf{w}}(n) \\ &+ \tilde{\mu}\mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{d}(n)\end{aligned} \tag{3–17}$$

其中 $\mathbf{I}$ 为单位矩阵。由上式(3–17)为了便于描述，定义投影算子 (projection operator) 如下：

$$\mathbf{P} \;=\; \mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{A}(n) \tag{3–18}$$

对方程(3–18)简单分析不难发现，$\mathbf{P}$ 由数据矩阵 $\mathbf{A}(n)$ 唯一确定。所以，对确定好了的 $\tilde{\mu}$,$\mathbf{A}(n)$ 和 $\mathbf{d}(n)$，互补投影算子 (complement projector) $[\mathbf{I}-\tilde{\mu}\mathbf{P}]$ 作用在前一个权重矢量 $\hat{\mathbf{w}}(n)$ 上，产生新的权重矢量 $\hat{\mathbf{w}}(n+1)$。最为重要的是，由于方程(3–17) 中第二项的存在，也就是 $\tilde{\mu}\mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{d}(n)$，使得互补投影算子成为了仿射投影算子而非简简单单的一个投影算子。

在参考文献[10] 的第八章中 (pp 385-429) 谈到，对于各种最小均方算法，我们可以证明，当 N 小于 M 时，矩阵 $\mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}$ 可以被视为数据矩阵 $\mathbf{A}(n)$ 的伪逆 (pseudoinverse) 矩阵。用符号 $\mathbf{A}^+(n)$ 来表示该伪逆矩阵，进一步将递推方程(3–17) 简化表示为

$$\hat{\mathbf{w}(n)} \;=\; [\mathbf{I}-\tilde{\mu}\mathbf{A}^+(n)\mathbf{A}(n)]\hat{\mathbf{w}}(n)+\tilde{\mu}\mathbf{A}^+(n)\mathbf{d}(n) \tag{3–19}$$

至此，基于仿射投影算法的自适应滤波器递推公式的相关已全部讨论。当然，参考文献[10] 中做了这样一个总结 — “正是由于递推公式(3–19)的定义，我们可以很直观地将基于仿射投影算法的自适应滤波器视作，介于归一化最小均方滤波器和递归最小均方滤波器之间的一种滤波器，就计算复杂度 (computational complexity) 和性能 (performance) 两方面来说。”

3.3 仿射投影自适应滤波器的稳定性分析

作出假设，即本文讨论的基于仿射投影算法的自适应滤波器的本质数学模型为多重回归模型 (multiple regression model)，由上节讨论和参考文献[10] 的最小均方滤波器部分 (pp 320-327) 可知，模型的数学表达式可以为，

$$\mathbf{d}(n) \;=\; \mathbf{w}^H\mathbf{u}(n)+\nu(n) \tag{3–20}$$

在方程(3–20)中，$\mathbf{w}$ 为模型的未知参数矢量，$\nu(n)$ 为加性干扰，在归一化最小均方滤波器中，估测权重矢量 (tap-weight vector) 是通过该滤波器数学模型后，计算得到的对真实 $\mathbf{w}$ 的一个估计。因此可以定义真实值和估测值之间的误差 (mismatch) 一个专门的矢量 —权重误差矢量 (weight-error vector) 如下，

$$\boldsymbol{\varepsilon}(n) \;=\; \mathbf{w} \;-\; \hat{\mathbf{w}}(n) \tag{3–21}$$

再由归一化最小均方滤波器的递推方程[10]

$$\hat{\mathbf{w}}(n+1) \;=\; \hat{\mathbf{w}}(n)+\frac{\tilde{\mu}}{\|\mathbf{u}(n)^2\|}\mathbf{u}(n)\mathbf{e}^*(n) \tag{3–22}$$

两式(3–21)、(3–22)相减，得

$$\boldsymbol{\varepsilon}(n+1) \;=\; \boldsymbol{\varepsilon}(n)-\frac{\tilde{\mu}}{\|\mathbf{u}(n)^2\|}\mathbf{u}(n)\mathbf{e}^*(n) \tag{3–23}$$

在参考文献[10, 11, 12] 中提到，设计归一化最小均方滤波器的一个核心思想便是让估测权重矢量在每一次 n 至 $n+1$ 的迭代 (iteration) 中，尽可能达到最小值。而这样一种设计的实现，而这样一种最小化的设想，却又受制于迭代本身。受上述过程的启发，我们不难想到用均方差 (mean square deviation) 来作为最小方差滤波器稳定性的评价标准。

$$\mathscr{D}(n) \;=\; E[\|\boldsymbol{\varepsilon}(n)\|^2] \tag{3–24}$$

至此，对基于仿射投影算法的自适应滤波器稳定性分析的准备工作告一段落。现在正式开始分析。

因为仿射投影滤波器也归纳在最小均方滤波器类别下，所以同该滤波器一样，我们对其的稳定性分析以均方差 $\mathscr{D}(n)$ 为评价标准，均方差的定义在式(3–24)。用含有未知权重矢量 (unknown weight vector) 的多重回归模型方程减去方程(3–16)，我们可以得到下式

$$\boldsymbol{\varepsilon}(n+1) \;=\; \boldsymbol{\varepsilon}(n) - \tilde{\mu}\mathbf{A}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n) \tag{3–25}$$

将上述权重误差矢量的迭代公式代入 $\mathscr{D}(n)$ 的定义中，对得到的式子进行重排和化简，有

$$\begin{aligned}\mathscr{D}(n+1) - \mathscr{D}(n) \;=\;& \tilde{\mu}E[\mathbf{e}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)] \\ & - 2\tilde{\mu}E\{Re[\boldsymbol{\xi}_u^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)]\}\end{aligned} \tag{3–26}$$

其中

$$\boldsymbol{\xi}_u(n) \;=\; \mathbf{A}(n)(\mathbf{w} - \hat{\mathbf{w}}(n)) \tag{3–27}$$

为非干扰误差向量 (undisturbed error vector)。从方程(3–26)中可以比较明显地看出均方差 $\mathscr{D}(n)$ 随着 n 的单调 (monotonically) 增加而单调减少 —假定步长参数 μ 满足条件

$$0 < \tilde{\mu} < \frac{2E\{Re[\boldsymbol{\xi}_u^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)]\}}{E[\mathbf{e}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)]} \tag{3–28}$$

同样不难发现，对于上式，归一化最小均方滤波器的 $\tilde{\mu}$ 限定条件[10](pp 324-325) 是其一个特例。再对方程(3–26)进行求导分析，得到最优步长 (optimal step-size) 的数学定义如下，

$$\tilde{\mu}_{opt} \;=\; \frac{E\{Re[\boldsymbol{\xi}_u^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)]\}}{E[\mathbf{e}^H(n)(\mathbf{A}(n)\mathbf{A}^H(n))^{-1}\mathbf{e}(n)]} \tag{3–29}$$

由上式已经可以得出足够多可用的信息，但是我们利用参考文献[10, 11, 12] 中提到的几条假设，并对其在本节的讨论中做出扩展，是我们可以在更加细小的范围内讨论 $\tilde{\mu}_{opt}$，以便看出规律。为了叙述得更加清楚，做出的假设 (assumption) 描述采用英文，即对参考文献中的假设做出适当修改后直接引用。

- **Assumption I.** *F 对于每一次的迭代，该矩阵 $\mathbf{A}(n)\mathbf{A}^H(n)$ 逆的波动大小，作为 $\tilde{\mu}_{opt}$ 的分子和分母，可以小到忽略不计。进而将 $\tilde{\mu_{opt}}$ 简化为*

$$\hat{\mu}_{opt} \;\approx\; \frac{E\{Re[\boldsymbol{\xi}_u^H(n)\mathbf{e}(n)]\}}{E[\|\mathbf{e}(n)\|^2]} \tag{3–30}$$

- **Assumption II.** 非干扰误差矢量 $\boldsymbol{\xi}_u(n)$ 与干扰误差（噪声）矢量不相关

$$\boldsymbol{\nu}^H(n) = [\nu(n), \nu(n-1), ..., \nu(n-N+1)] \tag{3–31}$$

- **Assumption III.** 输入信号 $u(n)$ 的频谱，在比每一个权重误差矢量 $\boldsymbol{\varepsilon}(n)$ 所占带宽还要宽的区间内，都是平坦的。据此我们可以做出以下近似

$$\begin{aligned} E[\xi_u^2(n)] &= E[|e^T(n)\mathbf{u}(n)|^2] \\ &\approx E[\|\boldsymbol{\varepsilon}(n)\|^2]E[u^2(n)] \\ &= \mathscr{D}(n)E[u^2(n)] \end{aligned} \tag{3–32}$$

应用以上三条假设，对方程(3–29)进行化简，有

$$\begin{aligned} \tilde{\mu}_{opt} &= \frac{E[\|\boldsymbol{\xi}_u(n)\|^2]}{E[\|\mathbf{e}(n)\|^2]} \\ &= \frac{\sum_{k=0}^{N-1} E[|\xi_u(n-k)|^2]}{\sum_{k=0}^{N-1} E[|e(n-k)|^2]} \\ &\approx \frac{\sum_{k=0}^{N-1} \mathscr{D}(n-k)E[|u(n-k)|^2]}{\sum_{k=0}^{N-1} E[|e(n-k)|^2]} \end{aligned} \tag{3–33}$$

第四章　信道评估方法设计

评价数字通信系统的各项指标中，接收到的信号与发送的信号之间出错的概率（即 Bit Error Rate,BER 误码率）是一项很重要也广泛被使用的评价指标。用蒙特卡洛方法进行通信系统信道评估，要点在于设计好一个大小为 M 的字母表（或者其他你想要进行测试的数据，比如二进制数据报）作为发送信号（即系统的输入信号）。参考文献 [2] 中指出通常设计的字母表大小为 2^k，这是由电子计算机的特性决定的。

而计算误码率的通常方法是任意多次取任意长（由准确性考虑应当取得较长，视 $M = 2^k$ 大小而定）的接收信号（即系统的输出信号），计算其平均的出错次数。其数学叙述大体如下：假设 N 为输入信号的数据量 (N 个二进制数)，$n(N)$ 为观测到的错误数，则出错概率 p 定义为 $p = \lim_{N\to\infty} \frac{n(N)}{N}$。对于前述的当字母表的大小 $M = 2$ 的时候，一个专有的名称为比特差错概率（误码率）。

本章和第五章所讨论的数字通信系统，其模型简化为

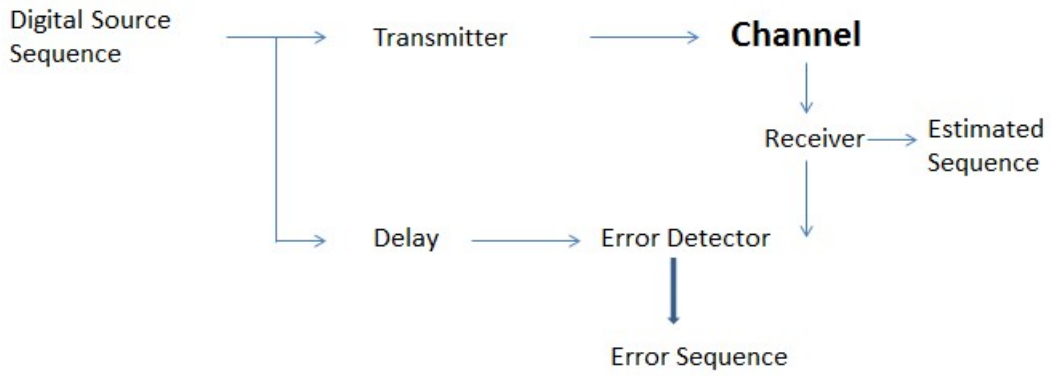

图 4–1: 本文分析采用的系统框图

4.1　蒙特卡罗方法

蒙特卡罗信道评估方法[2, 7] 是基于统计学方法的，对真实通信系统的一种模拟。蒙特卡罗方法的核心在于建立真实通信系统尽可能准确的数学模型，然后依据模型进行推理、演算。最为常见的模型建立方法是框图法 —一个例子如图4–1。根据任务要求，可以设计出类型相似但实际差别很大的框图出来[2, 7, 10]。如图4–2 和图??所示。

建立好所需系统的模拟框图之后，在模拟软件（本文采用的是[5]）里把每一个功能框都用对应的算法表示、代码实现出来。蒙特卡罗方法的一个好处是不需要知道

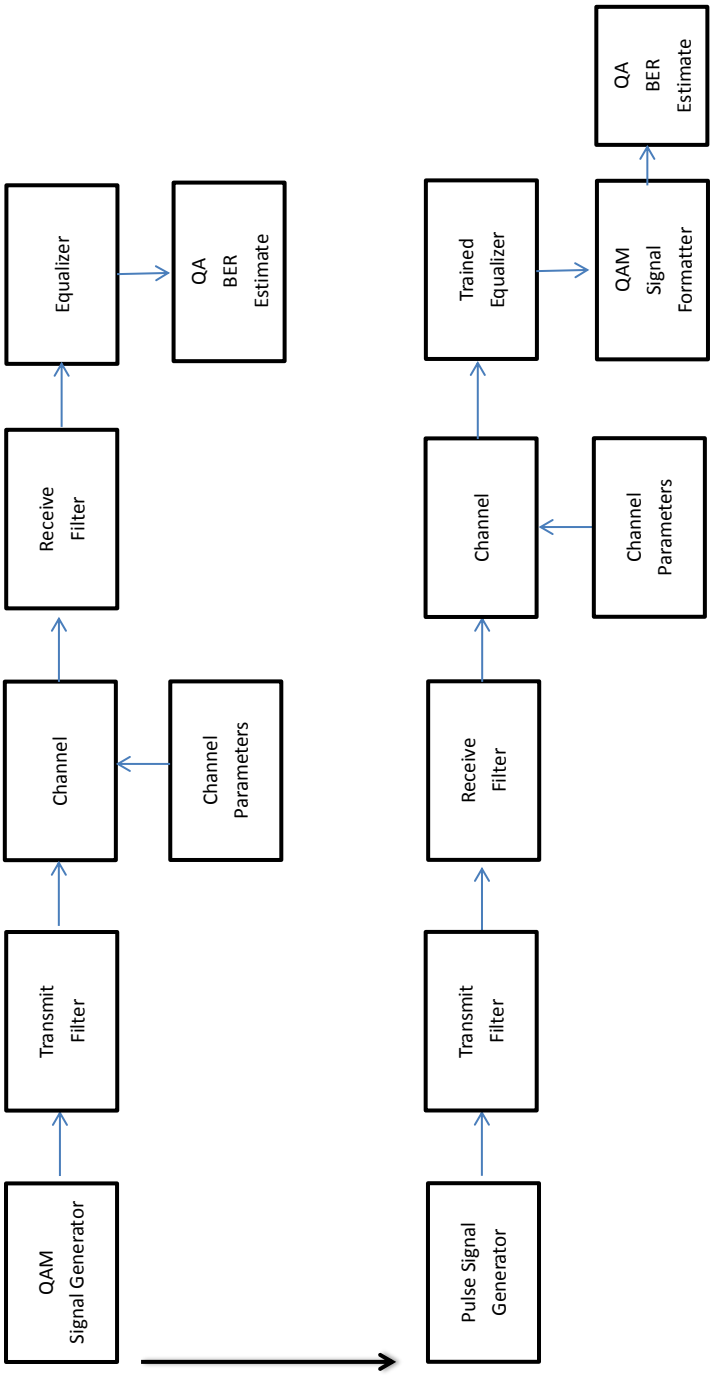

图 4–2: 一个复杂系统框图的例子

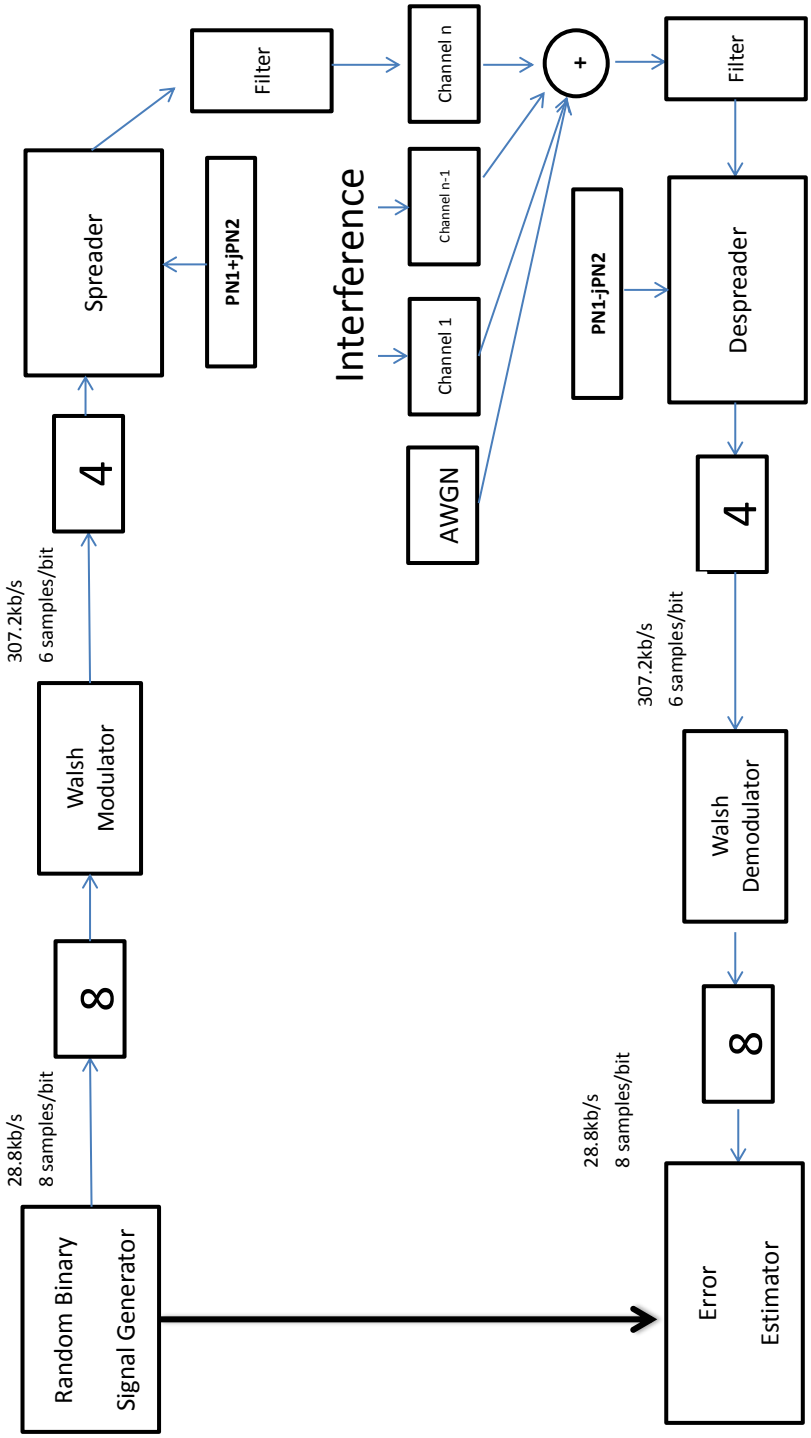

图 4–3: 另一个复杂系统框图的例子

系统真实的输入信号 —它将假定一个数字信号源作为输入。这样一来该方法便能很方便地比较输入和输出之间的误差，从而得出有关系统性能的若干结论。

对于本节讨论以及下一章部分计算采用的框图4–1来说，其中一个很重要的部件 (component) 就是系统的延时 (delay)。由参考文献[2, 7] 可知，讨论该系统（框图）的延时等同于讨论信号同步 (synchronization) 的问题。对于载波调制 (carrier-modulated) 的信号做输入量系统来说，信号解调的问题和信号同步的问题是一回事。限于本文主旨和篇幅，不做过多讨论。

4.2 准分析方法

准分析信道评估方法，也叫半解析 (semianalytical) 方法，是一种结合了系统模拟和纯数学推理计算的方法。这样一种方法可以让系统的输出信号几乎完全消除噪声的干扰[2]。该方法的描述如下，需要已知含有噪声的输出信号，并假定噪声是已知概率密度函数 (probability density function) 的加性噪声。由于在处理时假定噪声的概率密度函数已知，我们可以很方便地对输出端噪声的概率进行计算，这是数学推理的部分。在系统模拟部分，同样由于假定已知噪声的概率密度函数，我们可以在计算机模拟的时候人工移除噪声的影响（MATLAB 代码实现[13]）。一个简明的准分析方法如图 4–4 所示。该图的上半部分 (a) 表示的是一个假定的传输比特流，下半部分 (b) 表示的是去掉噪声后的接收机输入信号（即等待接收机进行处理的信号）。现结合准分析方法对该图进行讨论。如果用符号 ν_k 表示第 k 个采样瞬间对应的信号的幅值（即纵坐标的数值），当考虑加性噪声时，可以将每个采样瞬间误差的概率用方程4–1表示 [2, 4, 7]

$$P_{e_k} = \begin{cases} Prob[noise > \nu_k] = \int_{\nu_k}^{\infty} f_n dn, & \nu_k < 0 \\ Prob[noise < \nu_k] = \int_{-\infty}^{\nu_k} f_n dn, & \nu_k > 0 \end{cases} \tag{4–1}$$

其中 f_n 表示的是在处理中假定已知加性噪声的的概率密度函数，并且其平均值（即概率密度函数的数学期望）为了计算方便，假定为 0。这一假定使得在本次讨论中信号相关检测解调的阈值 (threshold) 也为 0。

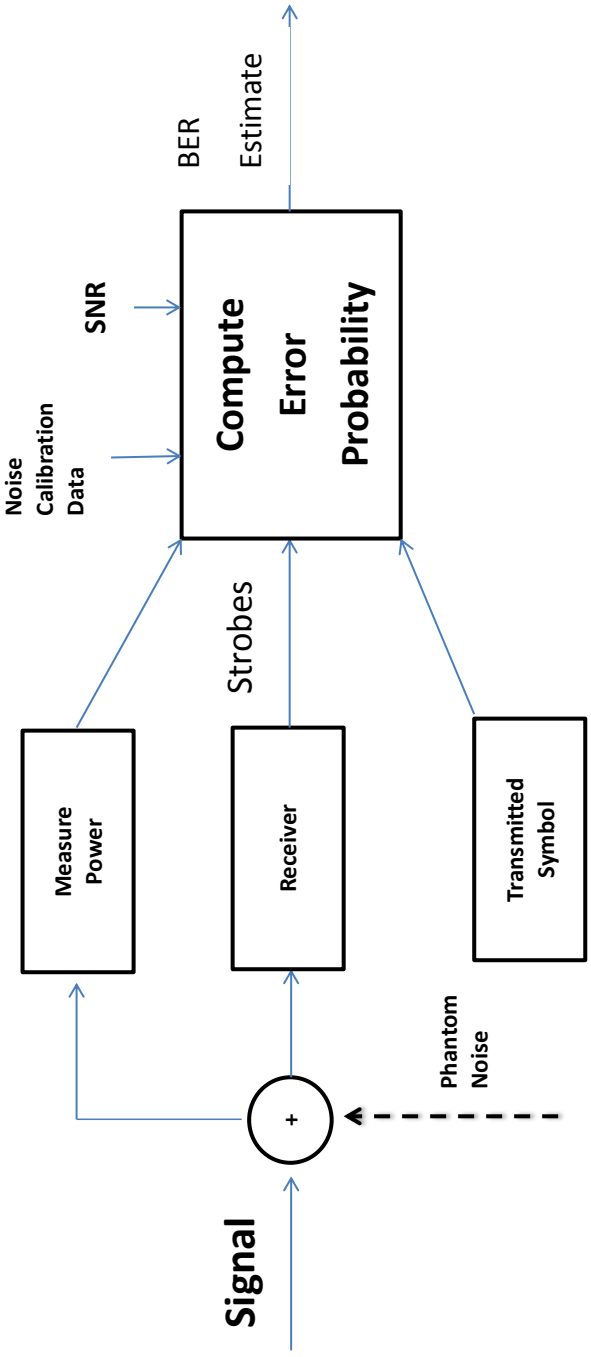

图 4–4: 准分析方法框图示例

第五章　信道评估结果及分析

本文仿真所用的代码由于较长，附在了附录里面。为了便于读者直观的分析、比较仿真结果，所得的一共 16 张图片均作为正文放在了本章。

5.1　仿真结果

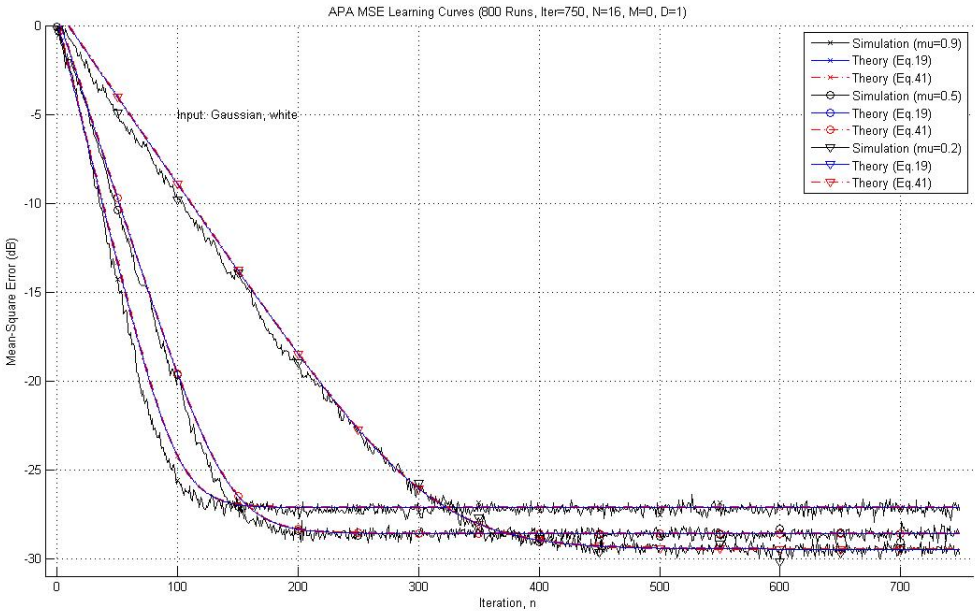

图 5–1: APA MSE Learning Curves, 800 Runs, Iter=750, N=16, M=0, D=1

5.2　结果分析

现在根据上节所列的仿真结果对全文做一个总结。基于仿射投影算法的自适应滤波器，在最小均方滤波器家族中有着比归一化最小均方滤波器更好的收敛程度，因此在计算复杂度不是主要考虑因素的前提下，仿射投影自适应滤波器是一个很好的选择 —如前所述，介于两个极端，归一化最小均方和递归最小均方两个极端之间。

- 仿射投影自适应滤波器的学习曲线包含有指数项求和的运算，因此有一定的计算复杂度。

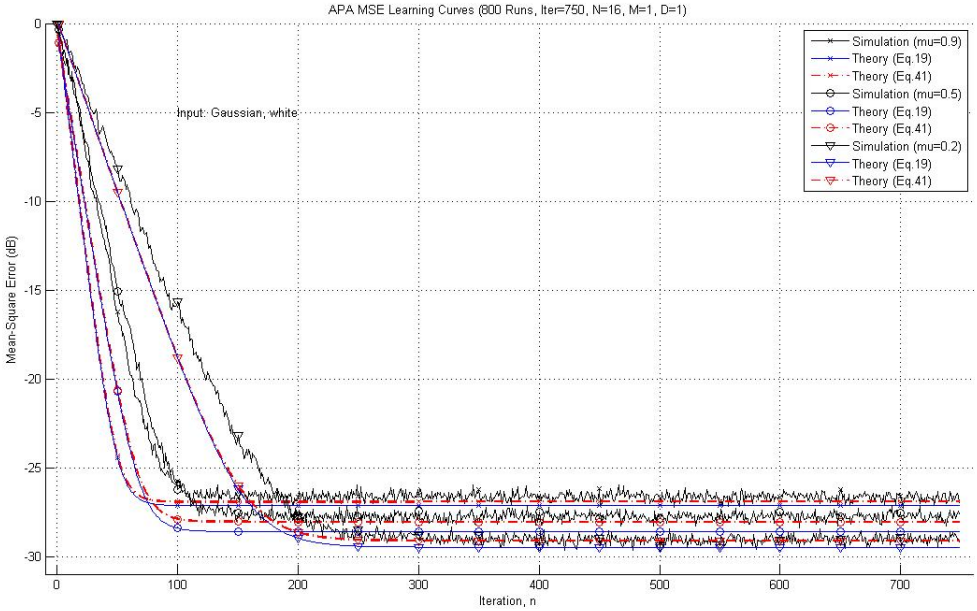

图 5–2: APA MSE Learning Curves, 800 Runs, Iter=750, N=16, M=1, D=1

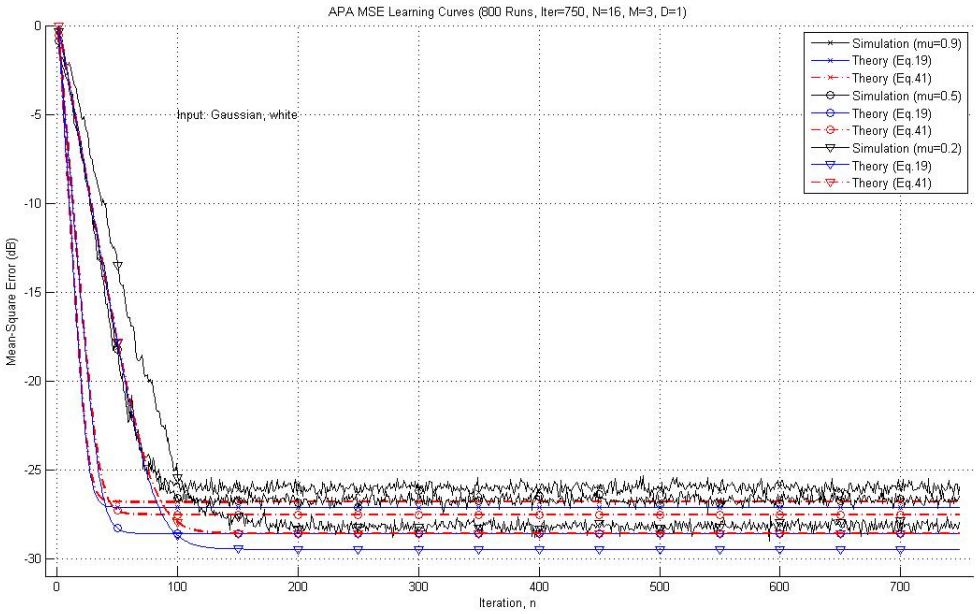

图 5–3: APA MSE Learning Curves, 800 Runs, Iter=750, N=16, M=3, D=1

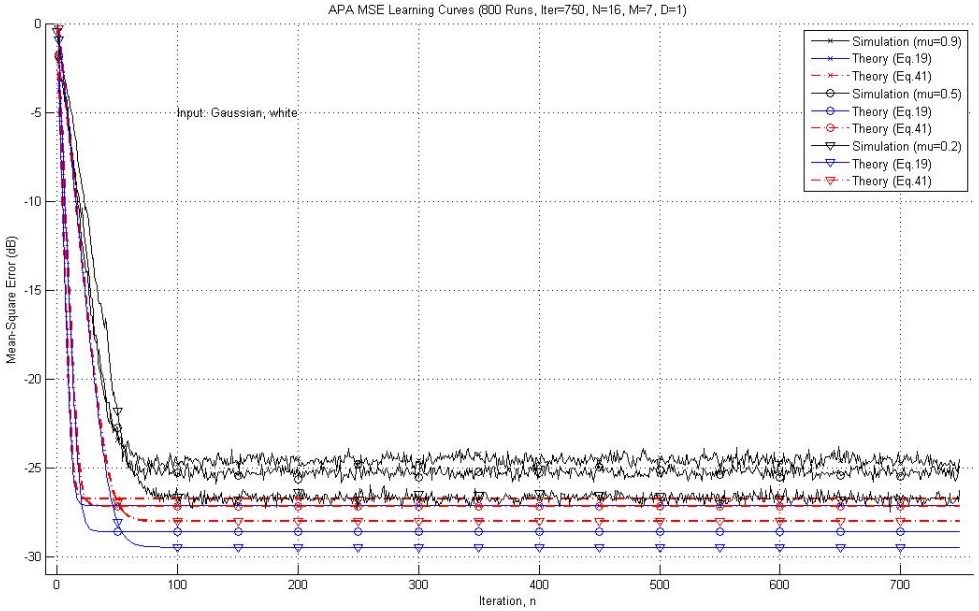

图 5–4: APA MSE Learning Curves, 800 Runs, Iter=750, N=16, M=7, D=1

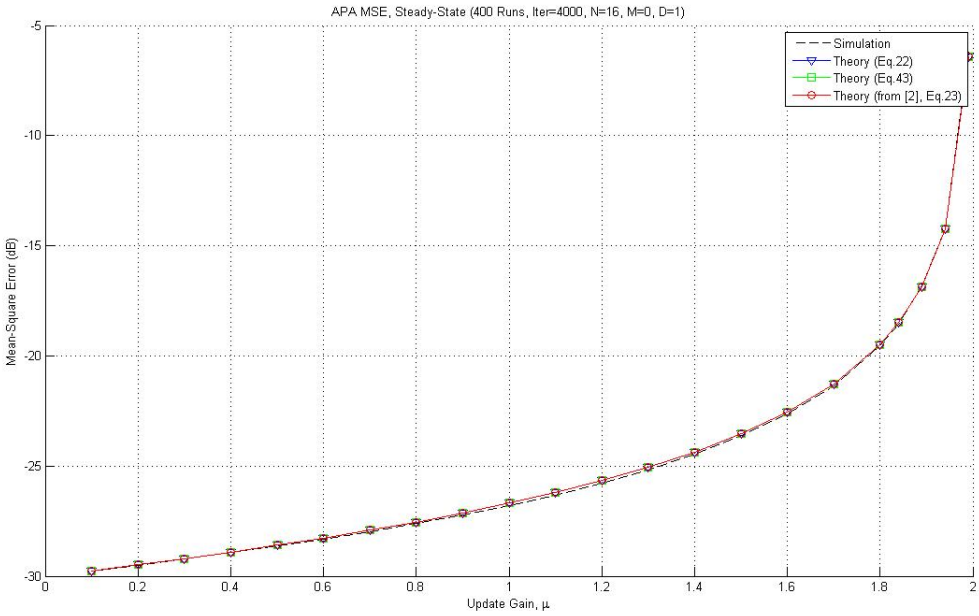

图 5–5: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=0, D=1

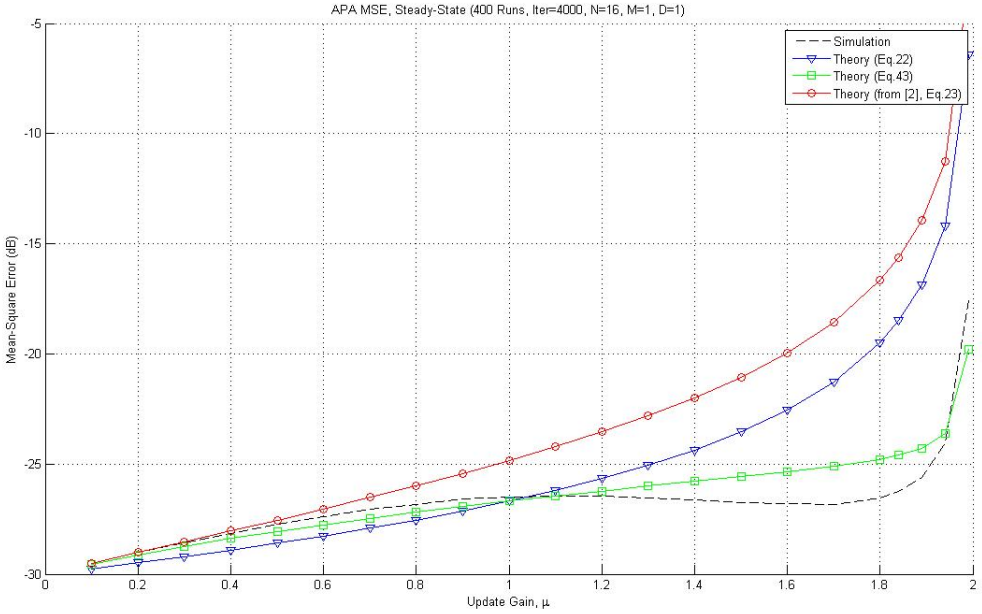

图 5–6: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=1, D=1

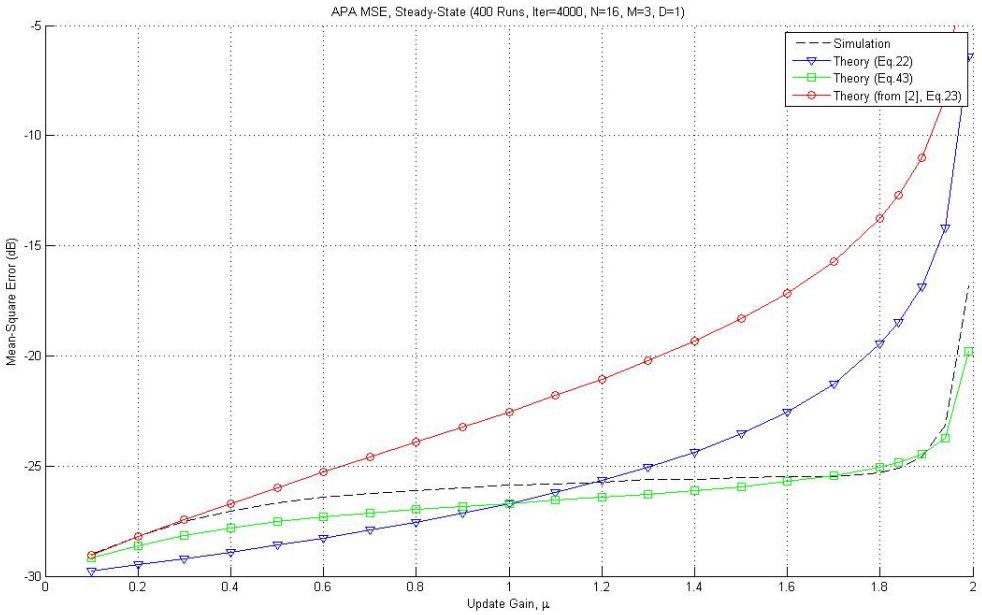

图 5–7: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=3, D=1

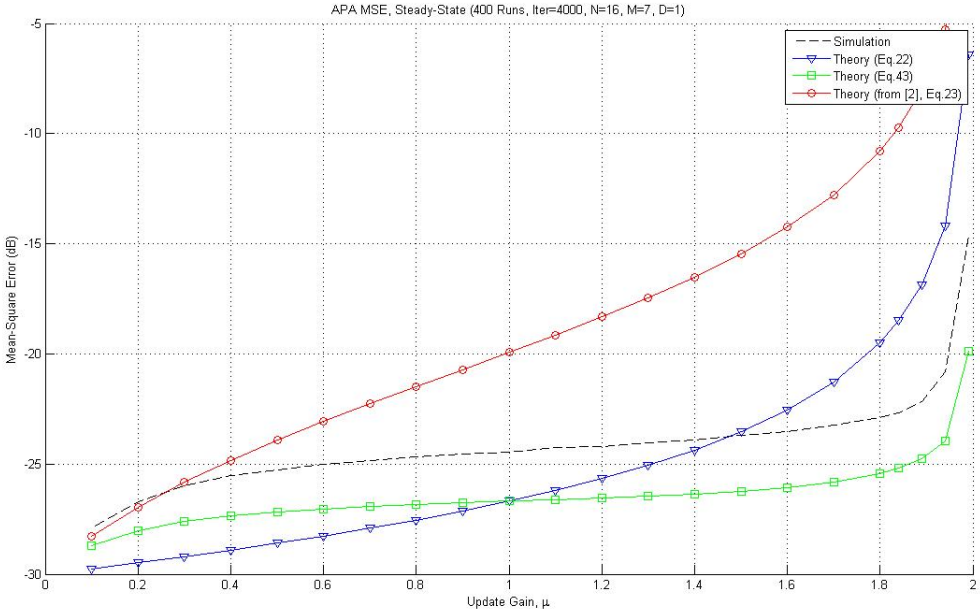

图 5–8: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=7, D=1

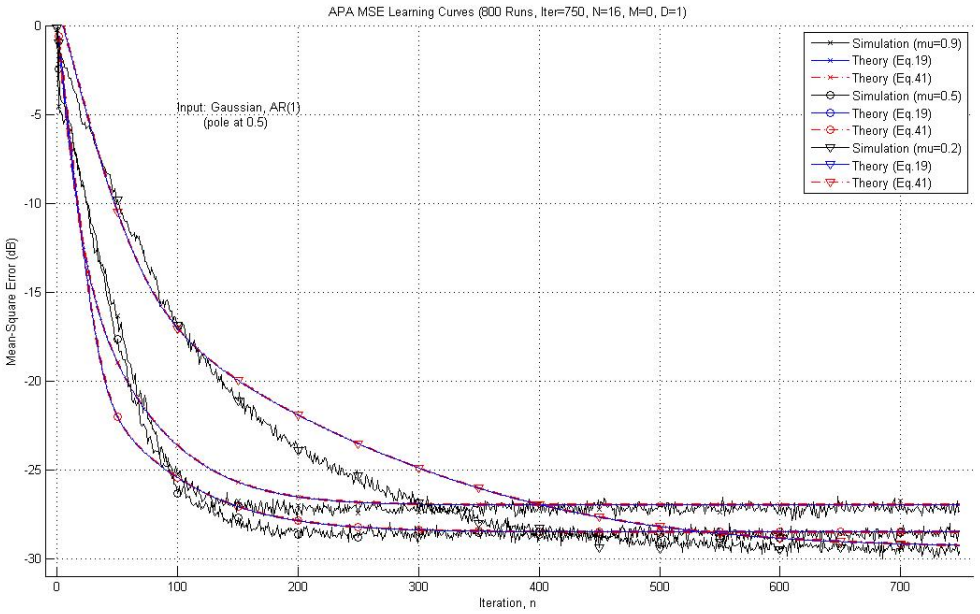

图 5–9: APA MSE Learning Curve, 800 Runs, Iter=750, N=16, M=0, D=1

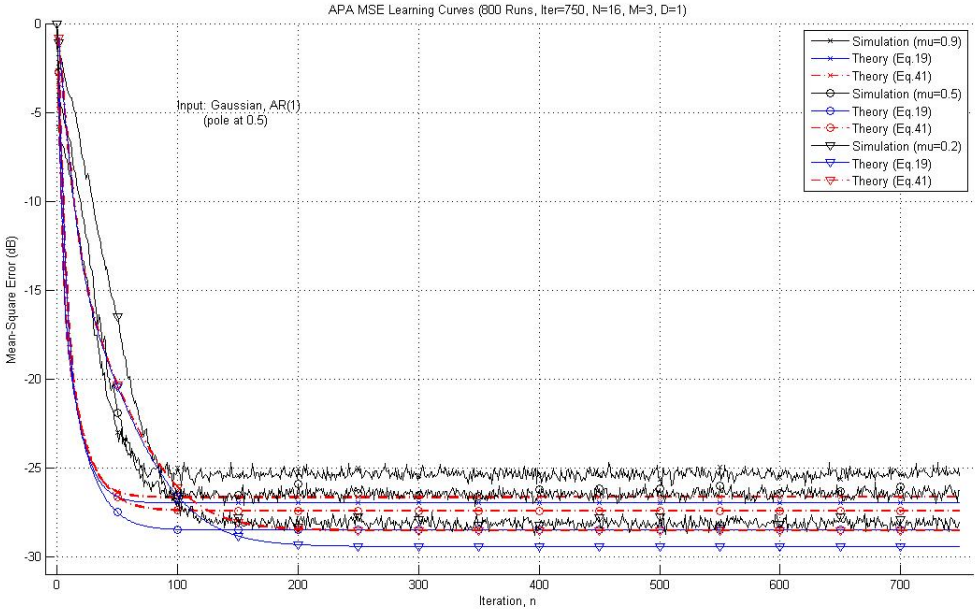

图 5–10: APA MSE Learning Curve, 800 Runs, Iter=750, N=16, M=3, D=1

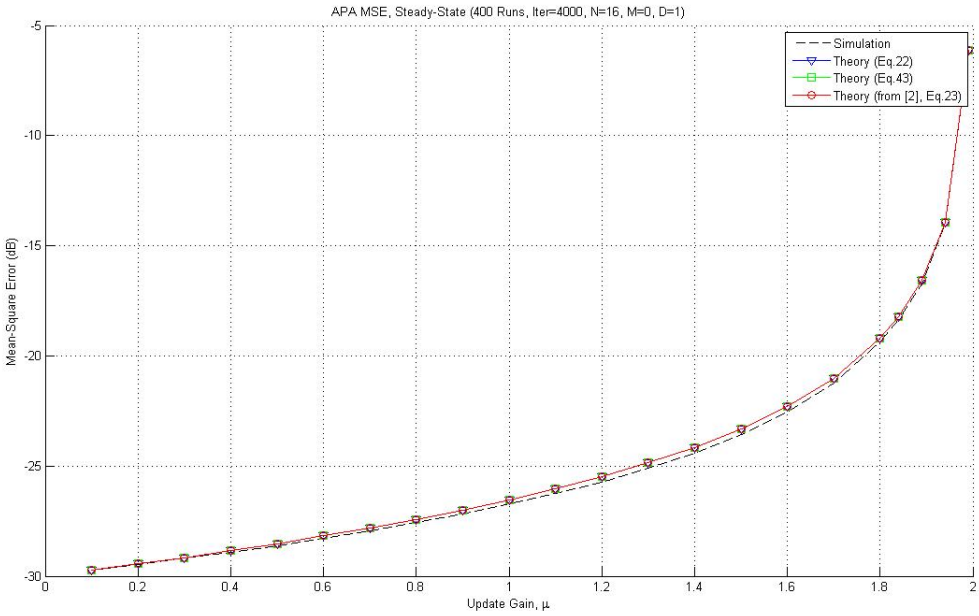

图 5–11: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=0, D=1

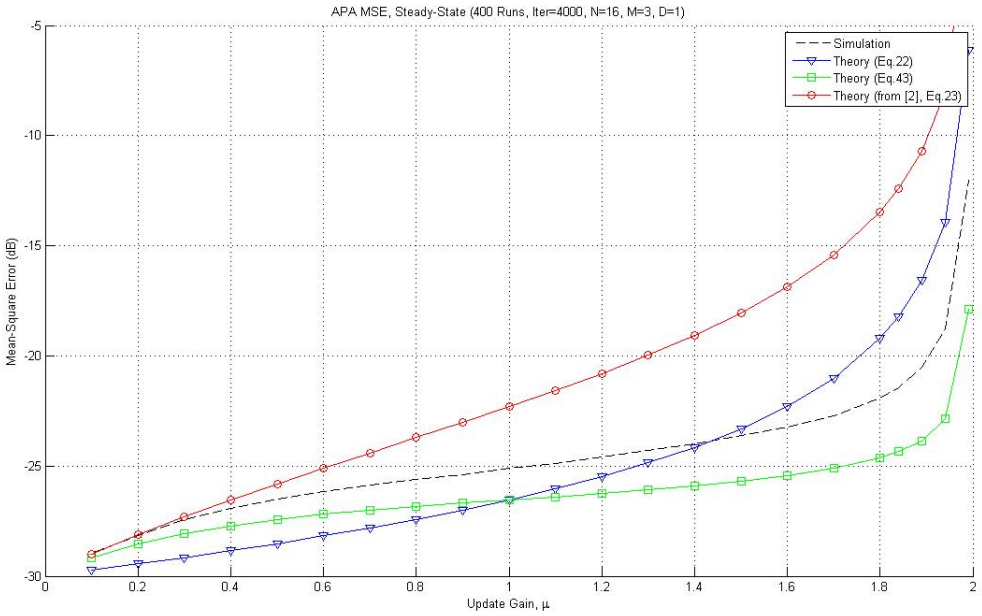

图 5–12: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=3, D=1

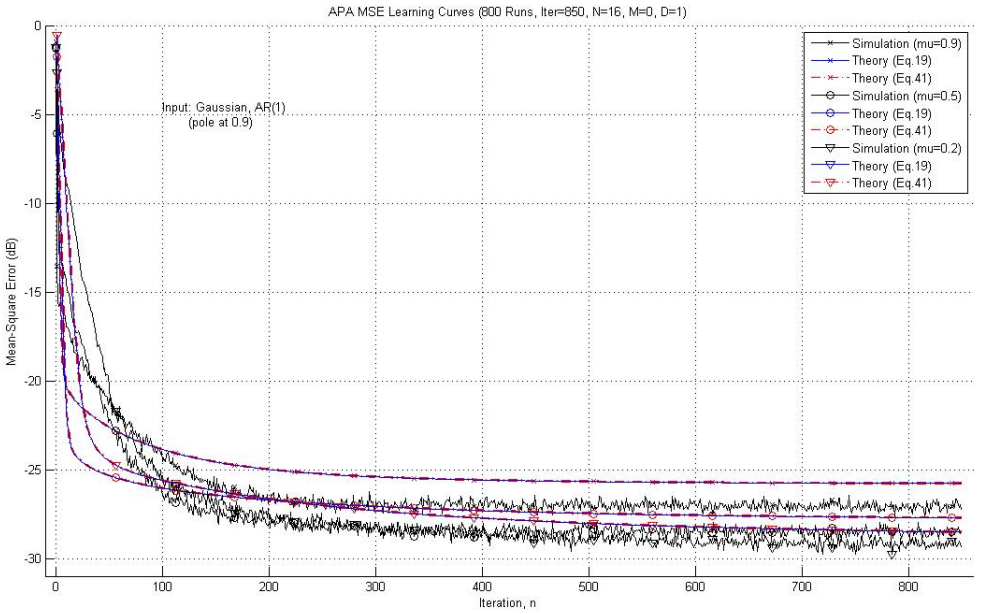

图 5–13: APA MSE Learning Curve, 800 Runs, Iter=850, N=16, M=0, D=1

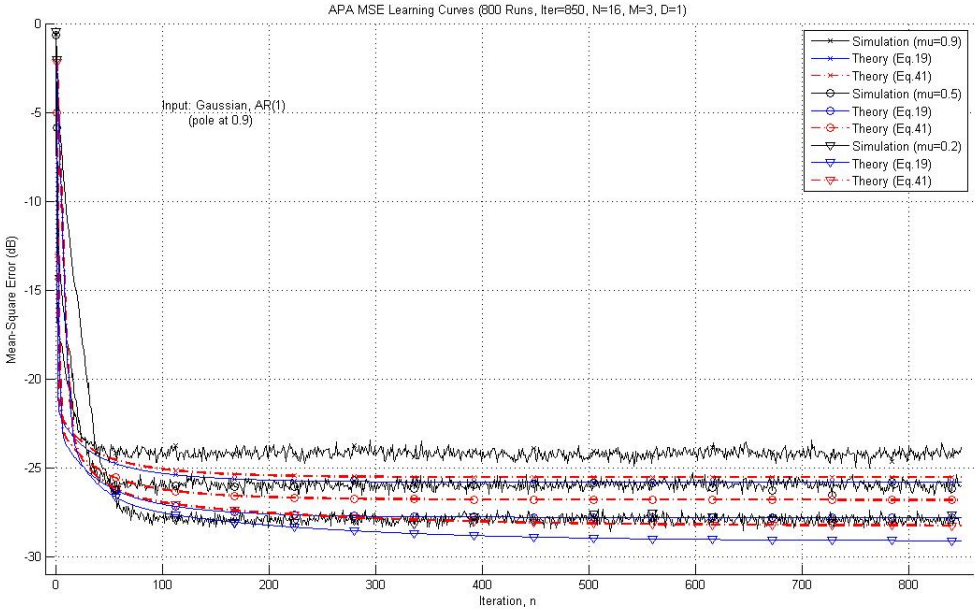

图 5–14: APA MSE Learning Curve, 800 Runs, Iter=850, N=16, M=3, D=1

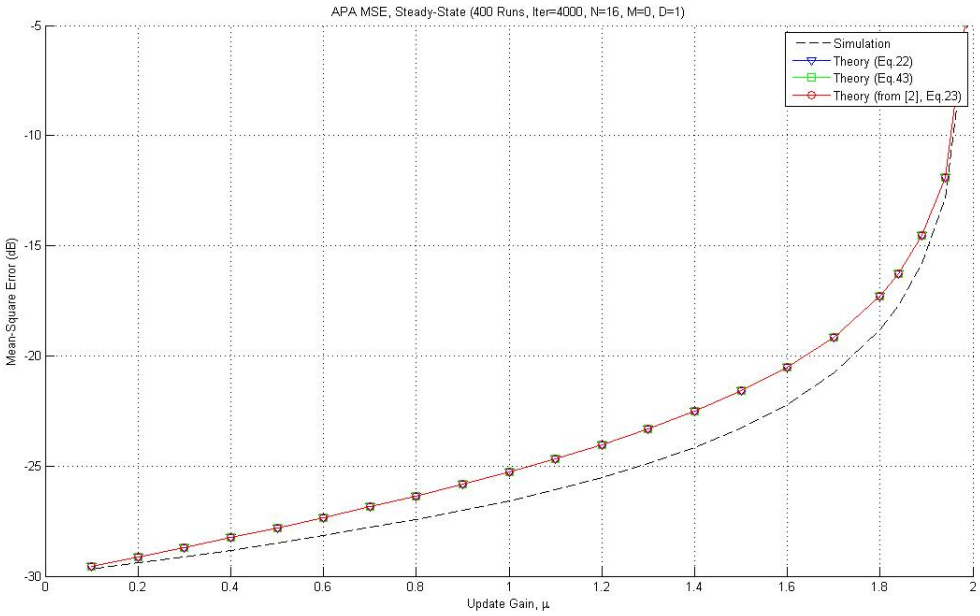

图 5–15: APA MSE Steady State, 400 Runs, Iter=4000, N=16, M=0, D=1

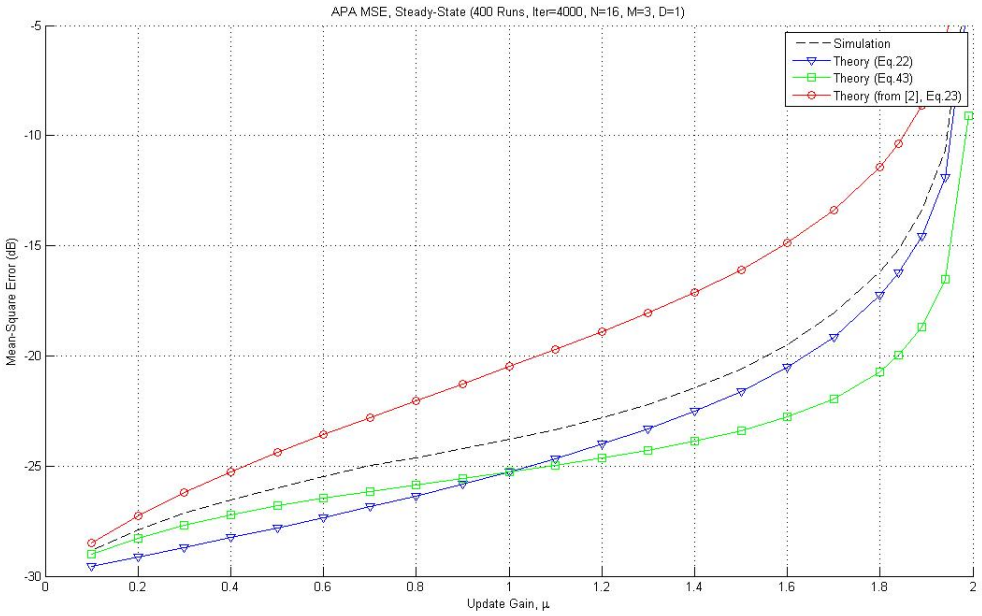

图 5–16: APA MSE Steady State, 400 Runs, Iter=400, N=16, M=3, D=1

- 仿射投影自适应滤波器的收敛速度比归一化最小均方滤波器要快，因此能经过较少次的迭代而获得期望结果。
- 当滤波器的阶数 N 增加的时候，在迭代时可以用的输入数据矢量 $\mathbf{u}_n$ 也随之增多。从理论和仿真两方面均证明了这样收敛性会变得更好，不过收敛速率却随之降低。

总的来说，通信系统的设计和评估领域有这样一句脍炙人口的话，可以很好地作为全篇的一个总结，“设计通信系统的时候没有免费的午餐可以吃，对于我们取得的每一分贝增益，我们都付出了等额的代价。”(*There is no free lunch in designing a communication system: for every gain that is made, there is a price to be paid.*)

参考文献

[1] Wikipedia. Eniac. Technical report, http://en.wikipedia.org/wiki/ENIAC, 2013.

[2] W. H. Tranter and K. L. Kosbar. Simulation of communication systems. *IEEE Communications Magazine*, pages 26–35, July 1994.

[3] J. M. Rabaey. The spice homepage. Technical report, Department of Electrical Engineering and Computer Science, 2013.

[4] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan. *Simulation of Communication Systems*. Springer, 2nd edition, October 2000.

[5] MathWorks. Matlab-the language of technical computing. Technical report, MathWorks, 2013.

[6] K. Ozeki and T. Umeda. An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properities. *Electronics and Communications in Japan*, 67-A(5):19–36, 1984.

[7] R. E. Ziemer, W. H Tranter, and D. R. Fannin. *Signals and Systems: Continuous and Discrete*. Prentice Hall, 4th edition, Febuary 1998.

[8] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 3rd edition, August 2009.

[9] A. V. Oppenheim, A. S. Willsky, and S. Hamid. *Signals and Systems*. Prentice Hall, 2nd edition, August 1996.

[10] S. Haykin. *Adaptive Filter Theory*. Pearson Education, Communications Research Laborotary, McMaster University, 4th edition, 2002.

[11] S. G. Sankaran and A.A. Beex. Convergence behavior of affine projection. *IEEE Transactions on Signal Processing*, 48(4):1086–1095, April 2000.

[12] S.G.Sankaran and A.A. (Louis) Beex. Fast generalized affine projection algorithm. *International Journal of Adaptive Control and Signal Processing*, pages 623–641, 2000.

[13] T. K. Paul and T. Ogunfunmi. On the convergence behavior of the affine projection algorithm for adaptive filters. *IEEE Transactions on Circuits and Systems*, 58(8):1813–1826, August 2011.

致　谢

本文从开题之初到设计实验、完成实验和论文撰写，及至今日在这里做最后的答辩工作，得到了很多同学与老师的帮助与支持。现一一感谢如下。

首先要感谢的就是我的指导老师智永锋。我在大四上学期完成所有课程之后就开始联系可能加入团队并进行毕业设计的老师们，譬如洪亮老师和张慧翔老师。在和智老师的沟通当中，他详细地介绍了所在课题组目前主力从事的一项国家自然科学基金研究，并表示我在该领域的研究成果将对日后的深造产生深远影响 —因为做的是很基础的东西。在毕业设计进行的过程当中，智老师坚持每周至少一次的会谈，对我工作的进度和遇到的困难嘘寒问暖。信号领域的基础研究，对数学的要求极高，在有我看不懂的公式的时候，智老师会带着我从头把公式推一遍，这也是为什么各位老师能够看到一篇有很多公式和推演的文章；由于我所在的专业对 MATLAB 实践能力的要求并不高，在我温习 MATLAB 的时候，有不懂的问题，一封电子邮件过去之后我过十几分钟就能收到很详尽的阐述，甚至附带了几篇相关的文章要求我阅读。智老师也是我出国留学推荐信的撰写人之一，能在大学的最后一段时间内把标准提到硕士研究生的水平，我想正是为我日后的科学研究道路在做准备吧！

本学期刚开学，我就向老师申请并顺利加入了自动化学院 511 教研室。在今年这一批研究生快要毕业的时候，我聆听了他们的预答辩，这给我准备现在的答辩提供了很大的帮助；在平时与研究生师兄师姐一起工作的时候，我能用我扎实的计算机科学功底帮助他们解决一些问题，也能从他们身上提早知道科研道路的艰辛与做出成果之后的开心。

在本文作者进行开题报告答辩的时候，唐伟教授和曲仕茹教授对本人提出的设计思路，依据他们多年科研领域的工作经验，进行了有效的质疑。这让本人在后面几个月少走了很多弯路，在此一并感谢。

本人算班级体里面的“捣蛋分子”了。在进行毕业设计、论文撰写的过程当中，偶尔由于进度和工作量的压力太大，对同学们进行了轻微的骚扰，他们也都表示理解和接受，一并感谢。

最后，但并不是最不重要的，我想感谢我的父母。大家都知道没有父母亲就没有我们这个道理，但是我是那种怎么着也找不到女朋友的孩子，因此为了克服每天对着不是书本就是笔记本（计算机）的苦楚，本人基本上每天要给家里去一个电话，跟爸爸或者妈妈谈谈心，汇报一下自己每天的工作和经验教训。在学术领域父母加起来可能还是没有我懂得多，也没有我在业务上这么熟练；但是在生活领域，我是彻头彻尾的小菜鸡，我想这也是同学眼力“学霸”不为人知的一面吧 —没有父母亲背后数十年如一日的支持，我想我早就退出科研领域，回老家安安稳稳过一辈子去了。

毕业设计小结

如果毕业设计的全过程仅仅是按照开题报告的要求，把该做的做出来，把该写的东西写好，那么我想教务部门也没有必要搞一个“小结”放在最后了，直接重新看一下我们的开题报告和摘要不就好了。他们需要用来评判的全部东西都在那里了。之所以需要我们撰写这一部分，我认为老师们是想了解更多幕后的事情。

真要写，这里也能写个成千上万字，毕竟是大学的最后一个学期了么，毕竟这个学期没有课程的安排，彻彻底底的回到时间由我们自由支配的“理想化状态”了。我在 2013 年 3 月份的时候去美国的一个网站申请了虚拟主机和自己的顶级域名 [1]，计划的就是从这个完全自由的学期开始把重要的东西记录下来，现在按照时间顺序和重要程度展示一点，我为自己的研究生生涯做的努力和准备吧。

- **二月**。与指导老师就毕业设计的选题进行了进一步的沟通。对即将进行为期 4 个月的研究做了很深入的背景调研，手段是搜索引擎 Google 和与国外教授的电子邮件。

- **三月**。在不少时间浪费掉之后，重新意识到时间管理与目标控制的重要性，花大力气阅读了”Getting Things Done, The Art of Stress-Free Productivity”，并因此很大程度提高了每天在教研室的工作效率。此外，还抽空阅读了《演讲之禅》与《黑客与画家》，对重新认识科研和商业领域有很大帮助。

- **四月**。在这个月集中攻坚了通信领域的理论基础。花很大的力气阅读并理解了”Signals and Systems”, ”Discrete-Time Signal Processing” 和”Principles of Communications” 三本麻省理工学院（MIT）的本科生教材。这打下了本文坚实的理论基础。

- **五月**。这个月从好友那里得知了一所很不错的只招研究生的学校的申请机会，并赶在截止日期之前与教授进行了沟通、做了申请。这个月的主要工作集中在设计前文所述的几个系统框图，并通过 MATLAB 编程对若干种设计思想进行了检验。

- **六月**。这个月完成了五月份开始的毕业设计论文撰写。并在指导老师智永锋的帮助下，对文章进行了修改和完善。此外，智老师校对了本人的英语翻译工作，还同样对答辩 PPT 提出了修改意见。

我的很多同学都选择了在国内继续接受硕士研究生的高等教育，我选择了一条更为艰辛的海外求学道路。正式接受 Offer 的学校是 *Masdar Institute of Science and*

[1] http://abrahamx.com

Technology，虽然在不为人知的阿布扎比，但是据一位今年正式毕业的系工大校友介绍，学风和校风都很好 —课程还很难。为此，我再一次感谢自己在这几个月里面养成的刻苦钻研、博学审问的习惯。而这种习惯对于一个立志在学术界做出一番成就的准研究生来说，正是不可或缺的。

附录 A 源代码

部分源代码重合程度较高，在此只列出核心的几个程序作为范例。

A.1 AffineProjection.m

```
function     [outputVector,...
              errorVector,...
              coefficientVector] =   Affine_projection(
                 desired,input,S)

%   Affine_projection.m
%   Implements the Complex Affine-Projection algorithm for
    COMPLEX valued data.
%
%   Syntax:
%       [outputVector,errorVector,coefficientVector] =
    Affine_projection(desired,input,S)
%
%   Input Arguments:
%       . desired   : Desired signal.
                                  (ROW vector)
%       . input     : Signal fed into the adaptive filter.
            (ROW vector)
%       . S         : Structure with the following fields
%           - step                  : Convergence (
    relaxation) factor.
%           - filterOrderNo         : Order of the FIR
    filter.
%           - initialCoefficients   : Initial filter
    coefficients.  (COLUMN vector)
%           - gamma                 : Regularization factor.
%                                     (small positive
    constant to avoid singularity)
%           - memoryLength          : Reuse data factor.
%                                     (referred as L in the
    textbook)
```

```
22 %
23 %   Output Arguments:
24 %       . outputVector      :   Store the estimated output
   of each iteration.   (COLUMN vector)
25 %       . errorVector       :   Store the error for each
   iteration.             (COLUMN vector)
26 %       . coefficientVector :   Store the estimated
   coefficients for each iteration.
27 %                               (Coefficients at one
   iteration are COLUMN vector)
28 %
29
30 %   Some Variables and Definitions:
31 %       . prefixedInput         :   Input is prefixed by
   nCoefficients -1 zeros.
32 %                                   (The prefix led to a
   more regular source code)
33 %
34 %       . regressor             :   Auxiliar variable. Store
    the piece of the
35 %                                   prefixedInput that will
   be multiplied by the
36 %                                   current set of
   coefficients.
37 %                                   (regressor is a COLUMN
   vector)
38 %
39 %       . nCoefficients         :   FIR filter number of
   coefficients.
40 %
41 %       . nIterations           :   Number of iterations.
42
43
44 %   Initialization Procedure
45 nCoefficients       =   S.filterOrderNo+1;
46 nIterations         =   length(desired);
47
48 %   Pre Allocations
49 errorVectorApConj       =   zeros((S.memoryLength+1)
```

```
,nIterations);
outputVectorApConj      =   zeros((S.memoryLength+1)
,nIterations);
coefficientVector       =   zeros(nCoefficients
,(nIterations+1));
regressor               =   zeros(nCoefficients
,(S.memoryLength+1));

%   Initial State Weight Vector
coefficientVector(:,1)  =   S.initialCoefficients;

%   Improve source code regularity
prefixedInput           =   [zeros((nCoefficients-1),1)
                             transpose(input)];
prefixedDesired         =   [zeros(S.memoryLength,1)
                             transpose(desired)];

%   Body
for it = 1:nIterations,

    regressor(:,2:S.memoryLength+1) =   regressor(:,1:S.
       memoryLength);

    regressor(:,1)                  =
    prefixedInput(it+(nCoefficients-1):-1:it);

    outputVectorApConj(:,it)        =
    (regressor')*coefficientVector(:,it);

    errorVectorApConj(:,it)         =
    conj(prefixedDesired(it+(S.memoryLength):-1:it))...
    -outputVectorApConj(:,it);

    coefficientVector(:,it+1)       =
    coefficientVector(:,it)+(S.step*regressor*...
    inv(regressor'*regressor+S.gamma*...
    eye(S.memoryLength+1))*errorVectorApConj(:,it));

end
```

```
88
89  outputVector              =    outputVectorApConj(1,:)';
90  errorVector               =    errorVectorApConj(1,:)';
91
92  %   EOF
```

A.2 AffineProjectionExample.m

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                          Example: System Identification
                                %
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %

                                %
5  %  In this example we have a typical system identification
       scenario. We want     %
6  % to estimate the filter coefficients of an unknown system
       given by Wo. In       %
7  % order to accomplish this task we use an adaptive filter
       with the same         %
8  % number of coefficients, N, as the unkown system. The
       procedure is:         %
9  % 1)  Excitate both filters (the unknown and the adaptive)
       with the signal       %
10 %   x. In this case, x is chosen according to the 4-QAM
       constellation.        %
11 %   The variance of x is normalized to 1.
                                                  %
12 % 2)  Generate the desired signal, d = Wo' x + n, which is
       the output of the     %
13 %   unknown system considering some disturbance (noise) in
       the model. The        %
14 %   noise power is given by sigma_n2.
                                                %
15 % 3)  Choose an adaptive filtering algorithm to govern the
       rules of coefficient %
```

```
%   updating.                                                            %
%                                                                        %
%      Adaptive Algorithm used here: Affine Projection                   %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%   Definitions:
ensemble     = 100;                          % number of realizations within the ensemble
K            = 500;                          % number of iterations
H            = [0.32+0.21*j,-0.3+0.7*j,0.5-0.8*j,0.2+0.5*j].';
Wo           = H;                            % unknown system
sigma_n2     = 0.04;                         % noise power
N            = 4;                            % number of coefficients of the adaptive filter
mu           = .1;                           % convergence factor (step)  (0 < mu < 1)
gamma        = 1e-12;                        % small positive constant to avoid singularity
L            = 1;                            % data reuse factor (L=0 -> NLMS; L=1 -> BNLMS; ...)


%   Initializing & Allocating memory:
W        = ones(N,(K+1),ensemble);   % coefficient vector for each iteration and realization; w(0) = [1 1 1 1].'
MSE      = zeros(K,ensemble);        % MSE for each realization
```

```
MSEmin  = zeros(K,ensemble);        % MSE_min for each realization


%   Computing:
for l=1:ensemble,

    X       = zeros(N,1);               % input at a certain iteration (tapped delay line)
    d       = zeros(1,K);               % desired signal

    x        = (sign(randn(K,1)) + j*sign(randn(K,1)))./sqrt(2); % Creating the input signal (normalized)
    sigma_x2 = var(x);
                                                % signal power = 1
    n        = sqrt(sigma_n2/2)*(randn(K,1)+j*randn(K,1));
             % complex noise

    for k=1:K,

        X       =   [x(k,1)
                     X(1:(N-1),1)];             % input signal (tapped delay line)

        d(k)    =   (Wo'*X(:,1))+n(k);          % desired signal

    end

    S   =   struct('step',mu,'filterOrderNo',(N-1),'initialCoefficients',...
                   W(:,1,l),'gamma',gamma,'memoryLength',L);
    [y,e,W(:,:,l)]  =   Affine_projection(d,transpose(x),S);

    MSE(:,l)    =   MSE(:,l)+(abs(e(:,1))).^2;
    MSEmin(:,l) =   MSEmin(:,l)+(abs(n(:))).^2;

end
```

```
68
69
70  %   Averaging:
71  W_av = sum(W,3)/ensemble;
72  MSE_av = sum(MSE,2)/ensemble;
73  MSEmin_av = sum(MSEmin,2)/ensemble;
74
75
76  %   Plotting:
77  figure,
78  plot(1:K,10*log10(MSE_av),'-k');
79  title('Learning␣Curve␣for␣MSE');
80  xlabel('Number␣of␣iterations,␣k'); ylabel('MSE␣[dB]');
81
82  figure,
83  plot(1:K,10*log10(MSEmin_av),'-k');
84  title('Learning␣Curve␣for␣MSEmin');
85  xlabel('Number␣of␣iterations,␣k'); ylabel('MSEmin␣[dB]');
86
87  figure,
88  subplot 211, plot(real(W_av(1,:))),...
89  title('Evolution␣of␣the␣1st␣coefficient␣(real␣part)');
90  xlabel('Number␣of␣iterations,␣k'); ylabel('Coefficient');
91  subplot 212, plot(imag(W_av(1,:))),...
92  title('Evolution␣of␣the␣1st␣coefficient␣(imaginary␣part)');
93  xlabel('Number␣of␣iterations,␣k'); ylabel('Coefficient');
```

A.3 AffineProjectionLearningCurve.m

```
1
2  %%% APA Algorithm for System Identification
3
4
5  % % %% number of runs, iterations...
6  % % num_runs=200;  num_iter=500;
7  % %
8  % %
9  % % %% Input type
```

```
% % GAUSSIAN_WHITE   = 1;
% % GAUSSIAN_COLORED = 2;
% % UNIFORM_WHITE    = 3;
% % UNIFORM_COLORED  = 4;
% % input_type = GAUSSIAN_WHITE; %UNIFORM_WHITE;
% %
% %
% % %% APA parameters
% % M = 16;     % input vector length
% % K = 4;      % number of past regressors, observ. to use
% % D = 8;      % delay between past regressors, observ. used
% % mu = 1.0;  % update gain parameter


%% Plant parameters
%v_var = 0.001;              % measurement noise (for 30dB)
epsil_I = 0;%0.001*eye(K);    % regularization parameter

% randomly generate optimum weights
w_opt = rand(M, 1);

%% For input signal statistics...
E_r2 = 0;
R = zeros(M,M);

%% For numerous runs of plant...
START_DLY = M + K*D;
wgt_vec = zeros(M,1);
mse_vec = zeros(num_iter,1);
for run_cnt = 1:num_runs

    %% Generate plant signal (before noise)
    if input_type == GAUSSIAN_WHITE
        u_i = randn(1, num_iter+START_DLY);         %
           gaussian, unit variance
    elseif input_type == UNIFORM_WHITE
        u_i = 2*rand(1, num_iter+START_DLY,1) - 1;  %
           uniform, between -1, +1
    elseif input_type == GAUSSIAN_COLORED
```

```
47          u_i = randn(1, num_iter+START_DLY);          %
               gaussian, unit variance
48          u_i = filter(1, [1 -0.9], u_i);              % pole
               at 0.9
49      else % UNIFORM_COLORED
50          u_i = 2*rand(1, num_iter+START_DLY,1) - 1;   %
               uniform, between -1, +1
51          u_i = filter(1, [1 -0.9], u_i);              % pole
               at 0.9
52  %       u_i = filter(1, [1 -0.5], u_i);              % pole
    at 0.5
53      end
54      y_i = filter(w_opt, 1, u_i);                     %
           noiseless plant output
55
56
57      % Find autocorr. R, E[1/r^2], for input signal, u_i
58      for i = START_DLY : (num_iter+START_DLY-1)
59          u_vec = u_i(i:-1:i-M+1).';
60          R = R + (u_vec * u_vec') / num_iter;
61          E_r2 = E_r2 + (1/(norm(u_vec)^2)) / num_iter;
62      end
63
64
65      %% Add measurement noise to plant output
66      scale_y = 1 / sqrt(mean(y_i.*conj(y_i)));
67      y_i = y_i * scale_y;                              %
           normalize plant power (x4)
68      v_i = sqrt(v_var)*randn(1, num_iter+START_DLY);  % noise
           for 30dB SNR
69      d_i = y_i + v_i;                                  % add
           noise to output
70
71      %% Initialize APA
72      w = zeros(M, 1);            % init. weights
73      U_i = zeros(K, M);
74      D_i = zeros(K, 1);
75
76      %% APA algorithm computation
```

```
77      for i = START_DLY : (num_iter+START_DLY-1)
78          % APA inputs (U_i and D_i)
79          for k = 0 : (K-1);
80              i_a = i - k*D;
81              i_b = i - k*D - M+1;
82              U_i(k+1,:) = u_i(i_a : -1 : i_b);
83              D_i(k+1,:) = d_i(i_a);
84          end
85
86          % Apply APA algorithm
87          Y_apa = U_i*w;
88          e_i = D_i - Y_apa;
89          w_p = w;
90          w = w + mu*U_i'*inv(epsil_I + U_i*U_i')*e_i;
91
92          % Store MSE output
93          mse_vec(i-START_DLY+1) = mse_vec(i-START_DLY+1) +
               e_i(1).*conj(e_i(1));
94
95      end
96
97      %% Add final weights to average
98      wgt_vec = wgt_vec + w;
99  end
100
101
102 %% Average MSE error, final weight  vector
103 mse_vec = mse_vec / num_runs;
104 wgt_vec = wgt_vec / num_runs;
105
106 % various parameters (R, E[1/r^2], etc...)
107 E_r2 = E_r2 / num_runs;
108 R = R / num_runs;
109 tr_R = trace(R);
110 [U,S,V] = svd(R);
111 p_v = diag(S)/tr_R;
112 %  p_v = ones(size(p_v))*(1/M);
113
114 % % %% Plot learning curves
```

```
115 % % figure; plot(0:num_iter-1, 10*log10(mse_vec));
116 % % title('APA MSE (Ensemble-averaged, 20 Runs, 1000 Iter.)
    ');
117 % % xlabel('Iteration, n'); ylabel('Mean-Square-Error, J(n)
    ');
118 % % axis([-10 510 -30 0]); grid on;
119
120
121 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122 format long;
123
124 % Display simulation MSE results
125 K__D__mu =[K D mu]
126 msevec_sim = mse_vec(end-3:end).'
                                          ;
127 msevec_avg = mean(mse_vec(end-floor(num_iter*0.2):end))
                 ;
128 E_r2
129
130 % Compute component for correl. of noise, coeff.
131 G = 0;
132 for i = 1:M
133     for g = 1 : (K-1)
134         G = G + p_v(i) * ( 1 - (1-(1-p_v(i))^g)/(1-(1-p_v(i)
            )^K) ) * (1-mu)^g;
135     end
136 end
137
138 % Compute theoretical MSE
139 mse_theory_a = v_var + (mu*v_var)/(2-mu)*E_r2*tr_R
                                     ;
140 mse_theory_b = v_var + (mu*v_var)/(2-mu)*E_r2*tr_R*(1+2*G)
                           ;
141 mse_theory_diniz = v_var + K*(mu*v_var)/(2-mu)*(1-(1-mu)^2)
    /(1-(1-mu)^(2*K))     ;
142
143
144 %%%%%%%%%%%%%%%%%%%%%%%%
145
```

```
146
147 % Theoretical MSE curve
148 a = mu*(2-mu);
149 lmb_ni_orig = zeros(M,num_iter);
150 lmb_ni_mod  = zeros(M,num_iter);
151 lmb_ni_orig(:,1) = diag(V'*w_opt*w_opt'*V)*scale_y^2;  %
    transformed (based on Rxx)
152 lmb_ni_mod(:,1)  = diag(V'*w_opt*w_opt'*V)*scale_y^2;  %
    transformed (based on Rxx)
153 mse_vec_theory_orig = zeros(num_iter,1);
154 mse_vec_theory_mod  = zeros(num_iter,1);
155
156 for n = 2 : num_iter
157     % find transformed covariance matrix
158     for i = 1:M
159
160         % beta term
161         b = 1-(1-p_v(i))^K;
162
163         % original formula...
164         lmb_ni_orig(i,n) = (1 - a*b)*lmb_ni_orig(i,(n-1)) +
               mu^2*v_var*E_r2*b;
165
166         % modified formula...
167         G_i = 0;
168         for g = 1 : (K-1)
169             G_i = G_i + ( (1-p_v(i))^g - (1-p_v(i))^K ) *
                   (1-mu)^g;
170         end
171         lmb_ni_mod(i,n)  = (1 - a*b)*lmb_ni_mod(i,(n-1)) +
               mu^2*v_var*E_r2*(b+2*G_i);
172     end
173     % compute theoretical MSE curves
174     mse_vec_theory_orig(n) = v_var + tr_R*sum(p_v.*
           lmb_ni_orig(:,n));      % assumes Rxx = I
175     mse_vec_theory_mod(n)  = v_var + tr_R*sum(p_v.*
           lmb_ni_mod(:,n));       % assumes Rxx = I
176 end
177
```

```
% Display results
msevec_theory_orig = mse_vec_theory_orig(end-3:end).'
                   ;
msevec_theory_mod  = mse_vec_theory_mod(end-3:end).'
                   ;

%pause

format short;
```

A.4 AffineProjectionSteadyState.m

```

%%% APA Algorithm for System Identification


% % %% number of runs, iterations...
% % num_runs=200;  num_iter=500;
% %
% %
% % %% Input type
% % GAUSSIAN_WHITE   = 1;
% % GAUSSIAN_COLORED = 2;
% % UNIFORM_WHITE    = 3;
% % UNIFORM_COLORED  = 4;
% % input_type = GAUSSIAN_WHITE; %UNIFORM_WHITE;
% %
% %
% % %% APA parameters
% % M = 16;     % input vector length
% % K = 4;      % number of past regressors, observ. to use
% % D = 8;      % delay between past regressors, observ. used
% % mu = 1.0;  % update gain parameter


%% Plant parameters
%v_var = 0.001;              % measurement noise (for 30dB)
epsil_I = 0;%0.001*eye(K);    % regularization parameter
```

```
27
28 % randomly generate optimum weights
29 w_opt = rand(M, 1);
30
31 %% For input signal statistics...
32 E_r2 = 0;
33 R = zeros(M,M);
34
35 %% For numerous runs of plant...
36 START_DLY = M + K*D;
37 wgt_vec = zeros(M,1);
38 mse_vec = zeros(num_iter,1);
39 for run_cnt = 1:num_runs
40
41     %% Generate plant signal (before noise)
42     if input_type == GAUSSIAN_WHITE
43         u_i = randn(1, num_iter+START_DLY);          %
            gaussian, unit variance
44     elseif input_type == UNIFORM_WHITE
45         u_i = 2*rand(1, num_iter+START_DLY,1) - 1;   %
            uniform, between -1, +1
46     elseif input_type == GAUSSIAN_COLORED
47         u_i = randn(1, num_iter+START_DLY);          %
            gaussian, unit variance
48 %       u_i = filter(1, [1 -0.9], u_i);              % pole
    at 0.9
49         u_i = filter(1, [1 -0.5], u_i);              % pole
            at 0.5
50     else % UNIFORM_COLORED
51         u_i = 2*rand(1, num_iter+START_DLY,1) - 1;   %
            uniform, between -1, +1
52         u_i = filter(1, [1 -0.9], u_i);              % pole
            at 0.9
53 %       u_i = filter(1, [1 -0.5], u_i);              % pole
    at 0.5
54     end
55     y_i = filter(w_opt, 1, u_i);                     %
        noiseless plant output
56
```

```
57
58      % Find autocorr. R, E[1/r^2], for input signal, u_i
59      for i = START_DLY : (num_iter+START_DLY-1)
60          u_vec = u_i(i:-1:i-M+1).';
61          R = R + (u_vec * u_vec') / num_iter;
62          E_r2 = E_r2 + (1/(norm(u_vec)^2)) / num_iter;
63      end
64
65
66      %% Add measurement noise to plant output
67      scale_y = 1 / sqrt(mean(y_i.*conj(y_i)));
68      y_i = y_i * scale_y;                                %
           normalize plant power (x4)
69      v_i = sqrt(v_var)*randn(1, num_iter+START_DLY);  % noise
           for 30dB SNR
70      d_i = y_i + v_i;                                    % add
           noise to output
71
72      %% Initialize APA
73      w = zeros(M, 1);            % init. weights
74      U_i = zeros(K, M);
75      D_i = zeros(K, 1);
76
77      %% APA algorithm computation
78      for i = START_DLY : (num_iter+START_DLY-1)
79          % APA inputs (U_i and D_i)
80          for k = 0 : (K-1);
81              i_a = i - k*D;
82              i_b = i - k*D - M+1;
83              U_i(k+1,:) = u_i(i_a : -1 : i_b);
84              D_i(k+1,:) = d_i(i_a);
85          end
86
87          % Apply APA algorithm
88          Y_apa = U_i*w;
89          e_i = D_i - Y_apa;
90          w_p = w;
91          w = w + mu*U_i'*inv(epsil_I + U_i*U_i')*e_i;
92
```

```
93          % Store MSE output
94          mse_vec(i-START_DLY+1) = mse_vec(i-START_DLY+1) +
                e_i(1).*conj(e_i(1));
95
96      end
97
98      %% Add final weights to average
99      wgt_vec = wgt_vec + w;
100 end
101
102
103 %% Average MSE error, final weight  vector
104 mse_vec = mse_vec / num_runs;
105 wgt_vec = wgt_vec / num_runs;
106
107 % various parameters (R, E[1/r^2], etc...)
108 E_r2 = E_r2 / num_runs;
109 R = R / num_runs;
110 tr_R = trace(R);
111 [U,S,V] = svd(R);
112 p_v = diag(S)/tr_R;
113 %  p_v = ones(size(p_v))*(1/M);
114
115 % % %% Plot learning curves
116 % % figure; plot(0:num_iter-1, 10*log10(mse_vec));
117 % % title('APA MSE (Ensemble-averaged, 20 Runs, 1000 Iter.)
        ');
118 % % xlabel('Iteration, n'); ylabel('Mean-Square-Error, J(n)
        ');
119 % % axis([-10 510 -30 0]); grid on;
120
121
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%
123 format long;
124
125 % Display simulation MSE results
126 K__D__mu =[K D mu]
127 msevec_sim = mse_vec(end-3:end).'                ;
128 msevec_avg = mean(mse_vec(end-999:end))          ;
```

```
129 E_r2
130 %p_v
131 R
132
133 % Compute component for correl. of noise, coeff.
134 G = 0;
135 for i = 1:M
136     for g = 1 : (K-1)
137         G = G + p_v(i) * (1-(1-p_v(i))^g)/(1-(1-p_v(i))^K) *
                (1-p_v(i))^(K-g) * (1-mu)^(K-g);
138 %%        G = G + p_v(i) * (1-(1-p_v(i))^g)/(1-(1-p_v(i))^K)
    * (1-mu)^(K-g);
139     end
140 end
141
142 % Compute theoretical MSE
143 mse_theory_orig_8 = v_var + (mu*v_var)/(2-mu)*E_r2*tr_R
                              ;
144 mse_theory_mod_23  = v_var + (mu*v_var)/(2-mu)*E_r2*tr_R
   *(1+2*G)                   ;
145 mse_theory_shin_23 = v_var + K*(mu*v_var)/(2-mu)*E_r2*tr_R
                            ;
146 mse_theory_diniz = v_var + K*(mu*v_var)/(2-mu)*(1-(1-mu)^2)
   /(1-(1-mu)^(2*K))    ;
147
148 format short;
```