高等学校教材

SIGNALS & SYSTEMS

信号与系统

—MATLAB综合实验

谷源涛　应启珩　郑君里

高等教育出版社

《信号与系统》(第二版)(上、下册)

郑君里 应启珩 杨为理 著
ISBN7-04-007981-X ISBN7-04-007983-6
2000 年 5 月
高等教育出版社

本书研究确定性信号经线性时不变系统传输与处理的基本概念和基本分析方法，从时域到变换域、从端口描述到状态空间描述，以实际应用为主要依据，兼顾离散和连续信号与系统之选材。本书在讲授传统内容的过程中充分体现时代气息，经典理论的论述与最新技术之引入密切融合，以当代信息科学的观点理解、审视、组织和阐述传统内容。注重结合基本概念，大量介绍各类应用实例（如PCM通信、CDMA通信、小波变换、宏观经济增长预测、人口模型等），强烈激发起学生的学习志趣和热情。第六章信号的矢量空间分析将正交、相关等重要概念以统一的数学和物理方法讲授，使学生对信号理论的学习步入更深层次，并初步理解它们的工程应用。这些内容的选取及其讲授方法为国内、外同类教材之首创，特色鲜明。

本书结构和选材有很大灵活性，能够适应电子工程与信息科学各类专业之需要。任课教师可根据实际情况选择不同章节组成深度和学时有区别的课程。目前至少有二百余所院校多种类型专业选用本书作教材，还有更多院校指定本书为研究生入学考试必读教程。

《信号与系统》学习辅导新书——

《教与写的记忆——信号与系统评注》

郑君里著
ISBN 7-04-017387-5
2005 年 8 月
高等教育出版社

本书是郑君里等著《信号与系统》(第二版)的配套教学辅导书，主要内容是对第二版原著各章作详细的评注解读，注重概念理解、分析对比、历史背景和综合应用。讨论题目源于作者数十年授课实践之积累（包括讲稿、讨论题、习题、考试题等综合资料）。对于协助教师备课、引导学生自学、考研综合复习以及准备海外留学都是相当理想的参考读物。

结合研究讨论，作者回忆了五十年学习与工作的感受，纵论国内外教学变革，其中对于清华园景物的描述，读来使人如临其境。即使对信号与系统不了解的读者，阅后也会感受到本书的独特魅力。

ISBN 978-7-04-022559-4
9 787040 225594 >
定 价 19.00 元

高 等 学 校 教 材

信号与系统

——MATLAB综合实验

谷源涛　应启珩　郑君里

高等教育出版社

内容简介

本书是郑君里等著《信号与系统》(第二版)的 MATLAB 上机实验配套教材。全书分七篇,其中第一、四、七篇集中讲授 MATLAB 编程技术,由浅入深逐步为其他各篇做好准备;第二、三、五、六篇按照连续时间信号与系统、离散时间信号与系统、通信系统和控制系统的顺序将 MATLAB 编程和信号与系统的综合复习密切融合,每篇最后安排一章综合性实例练习(音乐合成、语音合成、通信系统仿真和控制系统仿真)。

本书特点如下:首先,比较完整、系统地讲授 MATLAB 编程原理和技巧;其次,与原著密切结合,理论实践并重;最后,四个综合实验(大作业)对于激发学生的学习志趣和热情,提高分析问题和解决问题的能力有很大帮助。除紧密配合原著外,本书的练习题也具有自成一体的特点,因此可以作为各种"信号与系统"教材的实验辅助教材。同时可以为报考研究生的读者提供综合、深入复习的参考素材。

图书在版编目(CIP)数据

信号与系统:MATLAB 综合实验/谷源涛,应启珩,郑君里.
—北京:高等教育出版社,2008.1
ISBN 978-7-04-022559-4

Ⅰ.信… Ⅱ.①谷…②应③郑… Ⅲ.信号系统-计算机辅助计算-软件包,MATLAB Ⅳ.TN911.6

中国版本图书馆 CIP 数据核字(2007)第 186071 号

出版发行	高等教育出版社	购书热线	010-58581118
社　址	北京市西城区德外大街 4 号	免费咨询	800-810-0598
邮政编码	100011	网　址	http://www.hep.edu.cn
总　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　刷	北京明月印务有限责任公司	畅想教育	http://www.widedu.com
开　本	787×960　1/16	版　次	2008 年 1 月第 1 版
印　张	16.25	印　次	2008 年 1 月第 1 次印刷
字　数	300 000	定　价	19.00 元

物料号　22559-00

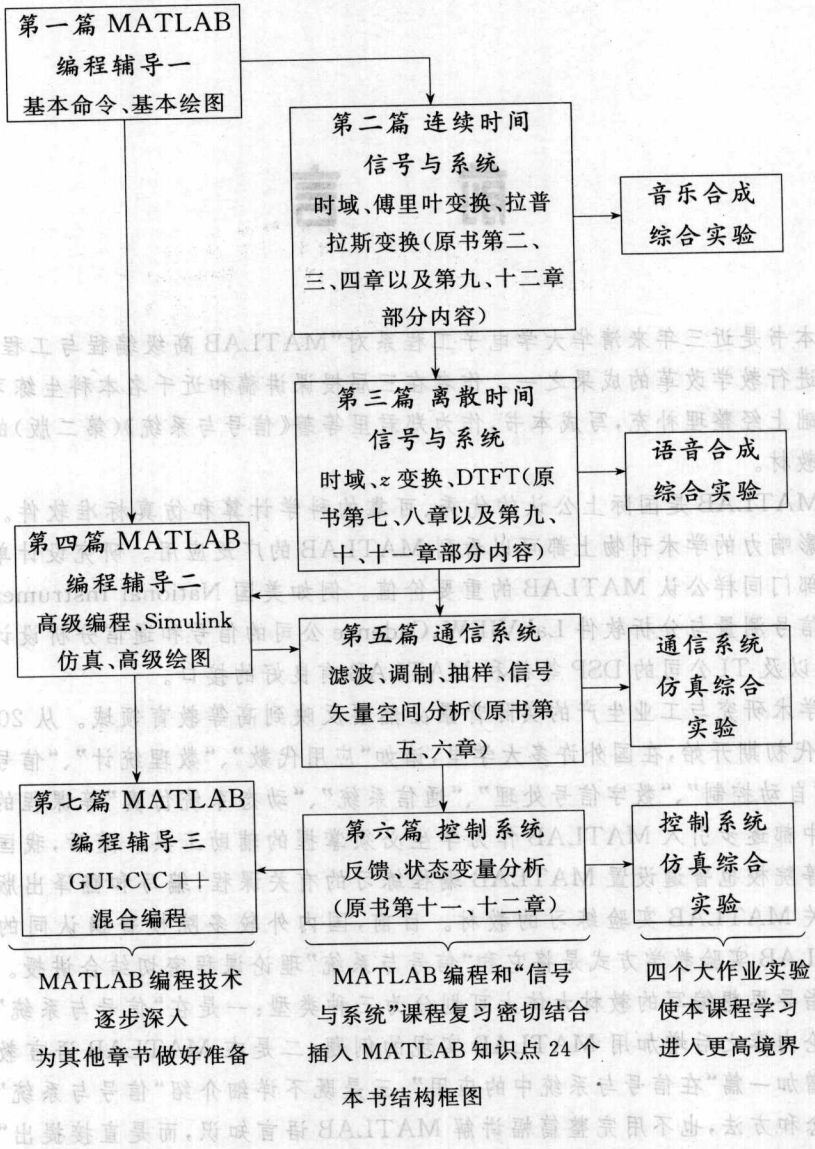
前　言

本书是近三年来清华大学电子工程系对“MATLAB高级编程与工程应用”课程进行教学改革的成果之一。作者在三届授课讲稿和近千名本科生练习实践的基础上经整理补充，写成本书，作为郑君里等著《信号与系统》(第二版)的配套实验教材。

MATLAB是国际上公认的优秀、可靠的科学计算和仿真标准软件。许多最具影响力的学术刊物上都可以看到MATLAB的广泛应用。研究设计单位和工业部门同样公认MATLAB的重要价值。例如美国National Instruments公司的信号测量与分析软件LabVIEW、Cadence公司的信号和通信分析设计软件SPW以及TI公司的DSP等都和MATLAB有良好的接口。

学术研究与工业生产的实际背景必然要反映到高等教育领域。从20世纪90年代初期开始，在国外许多大学里，诸如“应用代数”、“数理统计”、“信号与系统”、“自动控制”、“数字信号处理”、“通信系统”、“动态系统仿真”等课程的教学工作中都逐步引入MATLAB作为学生必须掌握的辅助工具。随后，我国的许多高等院校也普遍设置MATLAB编程练习的有关课程，编写和翻译出版了大量有关MATLAB实验练习的教材。目前，国内外较多院校普遍认同的引入MATLAB实验教学方式是将它和“信号与系统”理论课程密切结合讲授。按照这一指导思想编写的教材大体上可划分为三种类型：一是在“信号与系统”教材的理论内容之后增加用MATLAB实现的例题；二是在MATLAB语言教材的最后增加一篇“在信号与系统中的应用”；三是既不详细介绍“信号与系统”的基本概念和方法，也不用完整篇幅讲解MATLAB语言知识，而是直接提出“信号与系统”中的典型例题，再辅以MATLAB求解。本书的体系结构不同于以上三种类型，具有鲜明特色，显示全新风格。

下面以框图形式给出全书的基本内容和结构流程。可以看出，本书共包括七篇，按三条线索并行展开。首先考虑第一、四、七篇，这三篇集中讲授MATLAB编程技术，由浅入深逐步为其他各篇以及后续课程学习做好准备；其次是第二、三、五、六各篇将MATLAB编程练习和“信号与系统”课程的综合复习密切融合，学习这些内容可以获取、巩固编程技巧，综合深入掌握“信号与系统”基



本书结构框图

础理论知识，达到双重效果；第三，与第二、三、五、六篇对应，每篇安排一章综合性应用实例练习，这是四个相对独立的大作业，可以拓展视野、激发学习兴趣，使本课程学习进入更高境界。

由此容易理解，本书具有以下特色。

1. 比较完整、系统地讲授 MATLAB 编程原理和技巧。除了第一、四、七篇集中介绍这些内容外，在其他各篇结合“信号与系统”课程的应用实例循序渐进、逐步深入，引导读者掌握编程方法，并且穿插安排了“MATLAB 知识点”栏目共

24个，以利于分层次、全面深入学习编程知识。在组织这些材料时，力求内容简洁，努力协助读者在较短时间内掌握全面、适当的编程知识。另一方面，在第七篇将编程知识适当深入扩展，以利于和后续课程、科研工作及毕业设计的密切衔接。

2. 与原著密切结合，理论实践并重。郑君里等著《信号与系统》(第二版)教材的重要特色之一是例题和应用实例讨论丰富多彩，富有时代感。选择这些内容实现MATLAB编程具有得天独厚的优势，可以为授课教师安排作业、指导上机带来很大方便，同时也有利于学生自学并独立完成实验。本书中大部分例题选自主教材并注明对应的章、节、题号。此外，仍沿袭主教材风格补充了一些综合性、涉及多领域的编程例题，并提供了必要的背景知识(如音乐合成、语音合成等)。本书写作与原著配合默契，是一本相当理想的实验配套教材。当然，这些练习也具有自成一体的特点，因此可以成为其他各种“信号与系统”教材的实验辅助教材。同时可以为报考研究生的读者提供综合、深入复习的参考素材。

3. 综合实验(大作业)的独特功能。四个大作业——综合实验的安排为本书增添了活力和风采。由此产生的影响和效果绝不止局限于一门课程的改革。在本科教学培养的全过程中具有重要意义。我们注意到，在我国许多高校开设的“信号与系统”、“数字信号处理”、“随机过程”等课程与国外同类课程相比较，在理论内容方面并无明显差异，但在实验和大作业方面相对落后，迫切需要改进和加强。在此写入四个大作业之后为学生深入学习、拓宽知识面、开展研究型教学以及激发学习志趣和热情创造了十分有利的条件。对于提高学生分析问题和解决问题的能力将产生积极的作用。

回顾清华大学电子工程系本课程改革之历程可以看出，这种更新虽已初步取得成功，然而也曾经历曲折和困难，至今仍有一些问题有待改善。早在20世纪90年代中后期，我们编写了《MATLAB软件编程与应用》(内部教材)作为“信号与系统”课程的实验指导书，提供给学生做课外上机练习之用。多年来积累了一定的教学经验，也感受到明显的不足之处——实验学时太少、内容简单枯燥、很难激发学习兴趣。为了走出这一困境，2005年夏季小学期，我们首次对信息工程专业和电子与科学技术专业共约330名本科生开设“MATLAB高级编程与工程应用”课程，在刚刚结束的春季学期“信号与系统”64学时理论教学的基础上，为这门新课再提供32学时，其中讲授大约12学时，上机练习约20学时，大部分练习例题选自原书。这次改革取得了很好的教学效果，同学们普遍反映，在系统学习了这门新课之后，不但初步掌握了MATLAB语言，而且很好地复习总结了“信号与系统”理论课程。对于大作业的收获反响强烈。在学习后续“数字信号处理”、“通信原理”等课程中感觉更轻松，更有兴趣。

在编写本书过程中，感受到为写好大作业的背景知识，反复修改而经受的艰

辛历程。在教学实验过程中，也有同学害怕困难，不愿深入研究，但是较多同学还是努力克服困难，产生了强烈兴趣，收获显著。我们建议这些内容可以灵活安排，根据每个学生的能力和志趣因材施教，适当选做。

在2005年暑期新课程取得成功的基础上于2006年和2007年继续实施，每届学生人数和学时数都没有改变。而教学内容逐步丰富，并在同学们的支持下完成了本书的写作。

我们期望将本书奉献给各兄弟院校同类课程作为参考教材。同时考虑到各校具体情况的差异，有可能很难提供足够学时。因此，书中内容安排有较大灵活性，可以只选取第二、三、五、六各篇的基本内容提供上机练习。

考虑到计算机处理离散数据的特点，本书将主教材中稍后引入的离散时间信号与系统部分提前讲授，而把主教材中通信系统应用等内容移后。如果最后集中使用实验学时，当然这种安排不会在教学中引起麻烦。若将实验学时与理论教学穿插进行，必须注意本书第三篇的练习内容要适当推后，其他几篇也要陆续随之后移。

本书文字写作和MATLAB编程实验主要由谷源涛完成，应启珩、郑君里共同研究讨论全书结构和选材，参与部分写作并审阅了书稿。授课过程中与学生的密切交流对完成本书有很多重要的启发和帮助，特别要感谢姚卯青等同学提出的宝贵意见。

本书中涉及的网络资源请登录中国高校电工电子课程网获取，网址为http://ee.cncourse.com/electron/bookcenter/indexdefault。

限于水平及时间仓促，书中难免有不妥或错误之处，恳请读者指正，联系方式如下：

谷源涛（邮编100084；通信地址：北京市清华大学电子工程系；电话：010-62792782；电子信箱：gyt@tsinghua.edu.cn）

作者

2007年8月31日

于清华园

目　录

第一篇　MATLAB 编程辅导一　1

第一章　MATLAB 基础知识 …… 3
第一节　简介 …… 3
第二节　数据类型、数据结构、函数和命令 …… 5
第三节　数值、比较和逻辑计算 …… 7
第四节　符号计算 …… 9
第五节　M 文件——脚本和函数 …… 10
第六节　阶跃信号与冲激信号 …… 12
第七节　常用 MATLAB 命令 …… 13
第二章　MATLAB 绘图 …… 14
第一节　基本绘图操作 …… 14
第二节　句柄图形 …… 17
第三节　交互式绘图 …… 19

第二篇　连续时间信号与系统　23

第三章　连续时间系统的时域分析 …… 25
第一节　引言 …… 25
第二节　微分方程式的建立与求解 …… 28
第三节　零输入响应与零状态响应 …… 31
第四节　冲激响应与阶跃响应 …… 34
第五节　卷积 …… 37
第六节　小结 …… 40
练习题 …… 41

第四章 傅里叶变换 …… 42
第一节 傅里叶变换 …… 42
第二节 周期信号的傅里叶级数分析 …… 49
第三节 卷积特性(卷积定理) …… 52
第四节 小结 …… 55
练习题 …… 55
第五章 拉普拉斯变换、连续时间系统的 s 域分析 …… 57
第一节 拉普拉斯变换和逆变换 …… 57
第二节 系统函数(网络函数)$H(s)$ …… 61
第三节 由系统函数零、极点分布决定时域特性 …… 62
第四节 由系统函数零、极点分布决定频域特性 …… 64
第五节 二阶谐振系统的 s 平面分析 …… 67
第六节 小结 …… 68
练习题 …… 68
第六章 音乐合成 …… 70
第一节 背景知识 …… 70
6.1.1 乐音特征 …… 70
6.1.2 乐音基波构成规律 …… 70
6.1.3 乐音谐波的作用——音色 …… 72
6.1.4 乐音波形包络 …… 73
6.1.5 音调持续时间 …… 73
6.1.6 音符的叠接 …… 74
第二节 音乐合成综合实验 …… 74
6.2.1 简单的合成音乐 …… 74
6.2.2 用傅里叶级数分析音乐 …… 75
6.2.3 基于傅里叶级数的合成音乐 …… 76

第三篇 79

离散时间信号与系统

第七章 离散时间系统的时域分析 …… 81
第一节 常系数线性差分方程的求解 …… 81
第二节 离散时间系统的单位样值(单位冲激)响应 …… 84
第三节 卷积(卷积和) …… 85
第四节 解卷积(反卷积) …… 87
第五节 小结 …… 88
练习题 …… 89
第八章 z 变换、离散时间系统的 z 域分析 …… 90

第一节 z 变换定义、典型序列的 z 变换 …… 90
第二节 逆 z 变换 …… 91
第三节 利用 z 变换解差分方程 …… 93
第四节 离散系统的系统函数 …… 94
第五节 序列的傅里叶变换(DTFT) …… 96
第六节 离散时间系统的频率响应特性 …… 101
第七节 小结 …… 106
练习题 …… 107
第九章 语音合成 …… 108
第一节 背景知识 …… 108
9.1.1 发声机理 …… 108
9.1.2 语音信号的时域特征 …… 109
9.1.3 语音模型 …… 109
9.1.4 分析和合成语音 …… 112
第二节 语音合成综合实验 …… 114
9.2.1 语音预测模型 …… 114
9.2.2 语音合成模型 …… 118
9.2.3 变速不变调 …… 118
9.2.4 变调不变速 …… 119

第四篇 121

MATLAB 编程辅导二

第十章 高级编程知识 …… 123
第一节 函数和变量 …… 123
第二节 函数句柄 …… 125
第十一章 Simulink 仿真 …… 127
第一节 启动 Simulink …… 127
第二节 建立、打开和保存仿真模型 …… 128
第三节 编辑仿真模型 …… 128
第四节 运行仿真模型 …… 133
第五节 建立子系统 …… 134
第六节 利用 MATLAB 函数和程序 …… 135
第七节 访问工作空间中的变量和硬盘上的数据文件 …… 137
第八节 Simulink 支持的库和模块 …… 138
第十二章 高级绘图技术 …… 140
第一节 三维绘图和特殊图形 …… 140
第二节 图形高级控制 …… 145

第五篇 通信系统 147

第十三章 傅里叶变换应用于通信系统 …… 149
第一节 利用系统函数 $H(j\omega)$ 求响应 …… 149
第二节 无失真传输 …… 151
第三节 理想低通滤波器 …… 152
第四节 系统函数的约束特性 …… 154
第五节 调制与解调 …… 155
第六节 从抽样信号恢复连续时间信号 …… 162
第七节 脉冲编码调制(PCM) …… 165
第八节 小结 …… 167
练习题 …… 168
第十四章 信号的矢量空间分析 …… 169
第一节 相关 …… 169
第二节 能量谱和功率谱 …… 171
第三节 信号通过线性系统的分析 …… 171
第四节 匹配滤波器 …… 174
第五节 小结 …… 175
练习题 …… 176
第十五章 通信系统仿真 …… 177
第一节 背景知识 …… 177
15.1.1 频分多址(FDMA) …… 177
15.1.2 时分多址(TDMA) …… 177
15.1.3 码分多址(CDMA) …… 177
第二节 通信系统仿真综合实验 …… 178
15.2.1 FDMA 的 Simulink 仿真 …… 178
15.2.2 TDMA 的 Simulink 仿真 …… 180
15.2.3 CDMA 的 Simulink 仿真 …… 181
15.2.4 三种多址方式的比较 …… 183

第六篇 控制系统 185

第十六章 反馈系统 …… 187
第一节 引言 …… 187

16.1.1 控制系统工具箱中的 LTI 模型 …… 187
16.1.2 访问 LTI 模型的属性 …… 188
16.1.3 LTI 模型的组合 …… 189
第二节 反馈系统的基本特性及其应用 …… 191
16.2.1 改善系统频响特性 …… 191
16.2.2 使不稳定系统成为稳定系统 …… 193
第三节 根轨迹 …… 194
第四节 小结 …… 198
练习题 …… 199
第十七章 系统的状态变量分析 …… 200
第一节 状态矢量的线性变换 …… 200
第二节 系统的可控制性与可观测性 …… 203
第三节 小结 …… 205
练习题 …… 205
第十八章 控制系统仿真 …… 206
第一节 潜水艇下潜控制 …… 206
第二节 倒立摆平衡控制 …… 211

第七篇 219

MATLAB 编程辅导三

第十九章 图形用户界面(GUI)设计 …… 221
第一节 启动 GUI …… 221
第二节 设计和保存 GUI …… 223
第三节 运行 GUI …… 224
第四节 修改 GUI 控件属性 …… 224
第五节 编程控制 GUI 的方法 …… 226
19.5.1 OpeningFcn 函数 …… 226
19.5.2 回调函数(Callback Function) …… 226
19.5.3 访问控件 …… 226
19.5.4 控件之间数据共享 …… 227
19.5.5 访问工作空间中的数据 …… 227
第六节 本例的程序和运行结果 …… 227
第二十章 与 C/ C++混合编程 …… 230
第一节 准备工作 …… 230
20.1.1 配置 C/C++编译器 …… 230
20.1.2 理解 mxArray …… 231
第二节 从 MATLAB 中调用 C/C++程序 …… 232

20.2.1 C MEX 文件结构 …… 233
20.2.2 MEX API 函数 …… 233
20.2.3 C MEX 文件实例 …… 234
第三节 从 C/C++程序中调用 MATLAB 函数 …… 236
20.3.1 引擎库 API 函数 …… 236
20.3.2 程序实例 …… 236
20.3.3 用 VC++编译程序 …… 239
索引 …… 241
参考文献 …… 246

第一篇

MATLAB 编程辅导一

第一篇

MATLAB 编程指导

第一章 MATLAB基础知识

本章首先介绍MATLAB的基本知识，通过对界面的详细分析希望读者了解MATLAB的基本使用方法；然后从数据类型、数据结构、函数和命令入手，解释MATLAB的基本组成元素。从数值计算和符号计算两个方面举例讲解用MATLAB描述信号以及对信号进行运算的方法。接下来的两节介绍脚本和函数这两种M文件，并以阶跃函数和冲激函数为例做深入讲解。最后一节列出若干重要的函数和命令，同时介绍三个重要的函数集。

本章并不打算让读者对MATLAB有深入理解，只是希望对以后各章所需要用到的专项知识做基本的介绍，随着后续章节的学习，读者对MATLAB的理解会逐渐加强和深入。

第一节 简介

MATLAB是一门计算机程序语言，取名源于Matrix Laboratory，意在以矩阵方式处理数据。MATLAB将数值计算和可视化环境集于一体，并提供很多函数，功能强大且直观简便，因而应用范围非常广泛。一般认为MATLAB的典型应用包括：(1) 数值计算与分析；(2) 符号运算；(3) 建模与仿真；(4) 数据可视化；(5) 图形处理及可视化；(6) 基于图形用户界面的应用程序开发。这些内容在本书中均有涉及。

MATLAB由主包和数十个可选的工具箱组成。主包带有功能丰富和完备的数学函数库，大量复杂的数学运算和分析都可以直接调用MATLAB函数求解。工具箱为特定的学科和研究领域提供丰富的处理工具支持，本书主要涉及和“信号与系统”相关的若干工具箱，包括信号处理工具箱(Signal Processing Toolbox)、通信工具箱(Communication Toolbox)、控制系统工具箱(Control System Toolbox)等。

MATLAB启动后界面如图1.1所示。下面介绍其主要组成部分，通过这些介绍希望读者能够基本掌握MATLAB的使用方法。

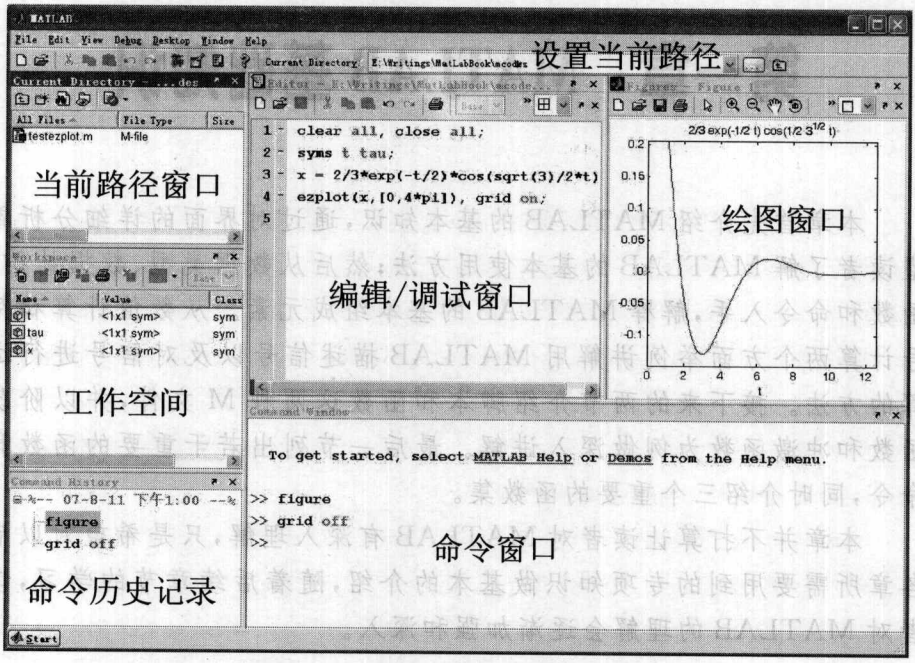


图1.1 MATLAB7.1用户界面

命令窗口(Command Window) MATLAB最基本的窗口，用户通过它来执行MATLAB命令。正常情况下提示符为“>>”，表示MATLAB进入准备工作状态。用户可在提示符后输入运算指令和函数调用等命令，MATLAB将迅速显示出结果并且再次进入准备工作状态。注意：如果命令后带有“;”，MATLAB执行命令但不显示结果。进入调试状态后，提示符将变成“K>>”。如果按上下键，MATLAB会按顺序依次显示刚才输入的命令，若要再次执行它，只要直接按回车键即可①。

命令历史记录(Command History) 保存并显示用户在命令窗口中输入过的命令，以及每次启动MATLAB的时间等信息。如果双击某条命令记录，则MATLAB会再次执行该命令。

工作空间(Workspace) 显示计算机内存中现有变量的名称、类型、结构及

① 如果开始按上下键时命令行上已有字符串，则顺序显示以该字符串开头的历史命令。

其占用字节数等。该窗口上有工具条支持用户将某变量存储到文件中或者从文件中载入某变量。如果直接双击某变量，则弹出 Array Editor 窗口供用户查看及修改变量内容。

编辑/调试窗口(Editor) 简单的运算可以在命令窗口中直接输入；若计算比较复杂，最好在文件中全部编辑好运算指令再统一执行，编辑/调试窗口就是MATLAB为此提供一个集成环境。该环境除了支持基本的编辑功能外，还提供解释执行和调试程序的功能，例如“设置断点(F12)”和“单步执行(F10)”等。MATLAB程序文件扩展名为“.m”，以文本文件形式保存。有两种方式运行程序文件：一是在命令窗口输入文件名；二是在编辑/调试窗口选择 Debug→Run 或按 F5 键运行程序。注意：对于不是通过 MATLAB 主界面打开的“.m”文件(比如在 Windows 资源管理器中双击打开)，编辑/调试窗口不支持调试功能。

图形窗口(Figure) 数据可视化是 MATLAB 强大功能的重要表现，图形窗口专门用来以图形方式显示数据。用户还可以编程或者手工修改各个图形元素的颜色、线型等属性。

设置当前路径(Current Directory) 用于选择当前工作路径。工作路径下的文件，可以通过在命令窗口中输入文件名的方式直接调用(只要在 MATLAB 的搜索路径列表中，都可以通过这种方式调用，搜索路径的概念后面再做详细介绍)。如果试图在编辑/调试窗口中运行一个不在搜索列表中的路径下的文件，MATLAB 会提示你是否要更改该路径为当前路径。

当前路径窗口(Current Directory) 显示当前工作路径内的所有文件，用户可以在这里新建或删除一个文件，也可以双击一个文件，在编辑/调试窗口中打开。

除上述窗口外，常见的 MATLAB 界面还包括帮助窗口、Simulink Library Browser 窗口、Simulink Model Editor 窗口、GUIDE 窗口、Profile 窗口等，本书后面在需要的地方会对这些界面做详细介绍。

学习 MATLAB 的最详细教材是它的帮助文件。请务必安装完整的帮助文档。获取帮助信息有两种方法：一是直接在命令窗口输入 help 函数名或命令；二是在帮助窗口中浏览或搜索相应信息。参考 MATLAB 的 Demo 程序也是学习编程的重要途径。

第二节 数据类型、数据结构、函数和命令

MATLAB 定义有数值(numeric)[①]、逻辑(logical)、字符(char)、符号(sym-

① 数值是多种数据类型的集合，包括整数型和浮点数型，整数型进一步可细分为有/无符号的单字节、双字节等，同样浮点数型可细分为单精度和双精度等，请由 help datatypes 了解详情。

ble)和函数指针(function_handle)等多种数据类型,同时定义有函数在多种数据类型之间进行相互转化①,以及判断某变量为何种类型的 isa 类函数②。大多数情况下,这些类型之间的转换是默认的,因而读者不必介意它们分别属于哪种类型,直接进行运算即可③。下面四条命令将 a、b、c、d 和 e 分别定义为数值型、逻辑型、字符型、符号型和函数指针型变量④。

```
a = 1
b = (1 == 1)
c = '1'
d = sym('1')
e = @sin
```

MATLAB 定义的数据结构包括矩阵、数组、单元数组(cell)和结构(struct)等。下面先介绍矩阵和数组两种类型⑤。

矩阵(标量和矢量是其特例)是 MATLAB 的基本数据结构,大部分运算都可以在矩阵意义下进行。矩阵用方括号"[ ]"进行定义,方括号内用逗号",",或空格分隔矩阵的列,分号";"或回车键分隔矩阵的行⑥。例如:

```
A = [a11,a12,a13;a21,a22,a23]
```

定义了一个 2×3 矩阵 A,其中每个元素 aij 是已赋值的变量或表达式。冒号":"用来构造一个元素为等差数列的行矢量,如命令

```
A = [0:2:8]
```

等价于

```
A = [0,2,4,6,8]
```

如果忽略两个冒号之间的项则默认增量为"1"。用 A(i,j)引用矩阵 A 的第 i 行第 j 列的元素。

从数据组织方式来看,数组和矩阵并无区别,但从运算角度两者截然不同,例如数组的平方是对其中的每个元素做平方。读者可以这样理解,如果不考虑运算,数组就等同于矩阵。

---

① 并非所有数据类型之间都可以相互转换,请由 help datatypes 了解详情。

② 可以调用 isa(x,'class_name')判断 x 是否为 class_type 类型,也可以调用 isdouble(x)直接判断其是否为双精度,请由 help isa 了解详情。

③ 个别类型,比如函数指针和其他类型之间,不可转换。

④ MATLAB 不需要对变量进行声明,可以直接定义使用。

⑤ 单元数组和结构两种类型的数据结构将在后文中需要的地方单独介绍。

⑥ 定义矩阵并非必须用方括号,用它只是为了避免分隔行列的逗号",","、";"空格或回车被错误理解为 MATLAB 命令之间的分隔符。例如,在命令窗口中输入"a=1"或者"a=1:5"同样将"a"定义为一个矩阵。

最后强调一点，矩阵和数组是数据结构，而数值、字符和符号等是数据类型，矩阵和数组中的元素可以是任何一种数据类型，但必须是同种数据类型[①]。

函数是MATLAB的基本功能单元，其调用方式为

```
函数名(参数1,参数2,...)
```

MATLAB建有完备的函数库为用户开发程序提供有力支持。命令的概念延续自字符界面的操作系统终端，比如

```
dir..
```

用于列出上一级路径下的所有文件，但在MATLAB中所有命令都可以函数形式调用，如

```
dir('..')
```

和上述命令的执行结果完全相同。本书中有时候混淆这两种称谓，因为事实上两者完全相同。

第三节 数值、比较和逻辑计算

MATLAB的数值计算以矩阵或数组为基本单元，定义了矩阵或数组之间四则运算、求幂和转置等。需要注意两点：第一，为了和矩阵运算相区分，数组运算的运算符的前面带一个“.”号，例如

```
c = a. * b
```

意味着a和b被当成数组结构，c中的每个元素等于a和b中对应位置的元素的乘积；第二，除法分为左除“\”和右除“/”，其中斜线的上方为被除数，下方为除数，例如已知Ax=y，则

```
x = A\y
```

表1.1列出了所有的MATLAB的数学运算符号。

表1.1 MATLAB中的数学运算符

名 称	说 明	名 称	说 明
+ -	矩阵加，矩阵减	*	矩阵乘
/ \	矩阵右除，矩阵左除	^	矩阵求幂
.* .^	数组乘，数组求幂	./ .\	数组右除，数组左除
' .'	共轭转置，转置	=	赋值

① 结构和单元中的元素可以是不同的数据类型。

MATLAB的比较和逻辑运算定义为对数组结构中的每个元素进行运算。逻辑值“真”和“假”分别用数值1和0表示，同数值量完全相同并可参与数值运算[①]。表1.2列出了所有的比较和逻辑运算符，其中all和any是两个有特色的逻辑函数。熟练运用比较和逻辑运算可以灵活巧妙的生成各种需要的变量。

表1.2 MATLAB中的比较和逻辑运算符

名 称	说 明	名 称	说 明
==	等于	~=	不等于
> >=	大于，大于等于	< <=	小于，小于等于
&	与	|	或
~	非	xor(a,b)	a和b异或
any(a)	a中有元素非零则为真	all(a)	a中所有元素都非零则为真

“信号与系统”中描述信号的基本方法是写出它的数学表达式，这对应于MATLAB中的符号表示法，将在下一节中介绍。在MATLAB中，描述信号的基本方法是数值表示法。此方法中连续时间信号和离散时间信号的界限已经消失，统一以抽样信号的形式用矢量表示，抽样间隔越小，信号连续性越强[②]。MATLAB对信号的描述和计算正是通过对矢量和矩阵的定义和操作实现的。下例演示用数值方法描述信号和对信号进行运算。

例1.1 已知 $x(t)=\sin(2\pi t)u(t)$，$y(t)=e^{-t}u(t)$。试计算 $t\in[-1,2]$ 区间的 $z_1(t)=2x(t)$，$z_2(t)=x(t-0.5)$，$z_3(t)=x(2t)$，$z_4(t)=x(t)+y(t)$，$z_5(t)=x(t)y(t)$。

解 完成本例的MATLAB命令如下，百分号%后为注释。

```
t = [-1:0.05:2];          %首先定义抽样间隔为0.05s的时间t
u = (t>0);                %然后用比较运算符定义阶跃信号u(t)
x = sin(2*pi*t).*u;       %定义x(t),pi是MATLAB预定义的常量
y = exp(-t).*u;           %定义y(t),exp和sin都是标准数学运算库
                          %函数
z1 = 2*x;                 %由数组或矩阵运算,可直接计算出z1、z4和
                          %z5
z4 = x+y;
```

① 在从数值量到逻辑量的强制转换中，0被转成“假”，非零被转成“真”。

② 矩阵用以表示多个同样长度的信号。

```
z5 = x. * y;
z2tmp = [zeros(1,0.5/0.05),x];      %求z2:先在x前加若干个0,对应于
                                    %延时0.5s
z2 = z2tmp(1:length(x));            %再截取前面的若干个抽样点,保证
                                    %t等长
z3 = x(1:2:length(x));              %求z3:直接从x中抽取出z3,但这
                                    %样做的抽样时间
t3 = t(1:2:length(t))/2;            %与样点数都和t不对应,所以新定
                                    %义t3和z3对应
```

例1.1中zeros(M,N)函数用于生成一个大小为M×N的全零矩阵,length(x)函数用于计算矢量x的长度。

第四节 符号计算

MATLAB中用var=sym(str)或syms var1 var2…定义符号变量。MATLAB支持的符号计算几乎和数值计算完全相同,表1.1中列出的运算符号同样可以针对符号变量进行。符号表达式到数值变量的转换可以用subs(f,x,y)完成,也就是用y替换掉表达式f中的x[①]。

最普遍的工程任务是对真实的抽样数据进行处理,因而数值法描述信号是工程专业的基础,也是MATLAB的传统优势。但从MATLAB5.3开始,MATLAB符号运算功能获得极大增强,到了MATLAB7.0,已经和符号运算专用计算语言Maple和Mathematic等的处理能力相差无几[②]。符号方法描述信号的优点是简单直观,理论性强,如下例。

例1.2 用符号计算的方法重做例1.1。

解

```
syms t x y z1 z2 z3 z4 z5           %声明说明符号变量
x = sin(2 * pi * t). * heaviside(t);  %定义x,注意heaviside(t)就是
                                    %u(t)
y = exp( - t). * heaviside(t);      %定义y
z1 = 2 * x;                         %直接计算z1、z4和z5
z4 = x + y;
```

① 如果y仍是符号变量,则subs实现符号替换。

② 但MATLAB的部分符号函数还是通过调用Maple的底层库实现的。

```
z5 = x. * y;
z2 = subs(x,t,t - 1);                    %用 subs 计算 z2 和 z3
z3 = subs(x,t,2 * t);
```

如果想计算出 t 取值从－1 到 2 时的 x 值，可以用

```
x1 = subs(x,t,[ - 1:0.05:2]);
```

计算其他函数值时方法相同，不再赘述。

第五节 M文件——脚本和函数

操作 MATLAB 的最简单途径是在命令窗口中直接输入命令行，以即时交互的方式编写程序。如果待处理的问题复杂且容易出错时，单纯使用命令行的方法会严重地降低工作效率。此时就需要将一行行的命令写在文件中，即“M文件”。MATLAB 是一门解释性的语言，因而 M 文件本身不能运行，必须有 MATLAB 环境的支持①。

M 文件分为脚本(Script)和函数(Function)两种。这两种文件都可以在编辑/调试窗口中运行，也都可以在命令窗口中输入文件名执行。事实上，MATLAB 工具箱中的函数就是一个个的 M 文件，它们和用户自己编写的 M 文件完全相同②。M 文件的文件名对字母大小写敏感，联机帮助中看到的都是大写③，但调用时都需用小写。

脚本式 M 文件是用户在命令窗口中输入的命令的简单集合，它的运行效果和用户在命令窗口中逐一的输入命令完全相同。脚本文件中定义的变量都是全局变量，脚本运行后它们会出现在(当前地)工作空间中，同时脚本也可以访问工作空间中已有变量。所以很多脚本文件(包括本书的例题)都以这样一行开始

```
clear all,close all,clc
```

即清空工作空间中的已有变量，防止对即将运行的程序带来负面影响，同时关闭所有已打开的窗口，清空命令窗口。

函数文件可以自带参数和返回值，因而比脚本文件复杂。函数文件的第一行用以声明这是一个函数文件，并且指定函数名、参数和返回值，即

```
function rvalue = functionname(param1,param2,...)
```

---

① 在初次运行 M 文件时，MATLAB 将其编成代码载入内存，因而执行速度很慢，但再次运行时，系统会从内存中直接取出代码运行，速度大大加快。这就是为什么第一次执行某条不常使用的命令时速度很慢的原因。

② 除非有特殊目的，否则不要把自己定义的文件名起得和标准函数名一样。

③ 只是为了醒目。

一般取 functionname 和该函数文件的文件名相同①。

函数文件中的变量都不是全局变量，不会在工作空间中出现，也不能被其他文件，比如脚本文件使用。同时，函数文件也无法访问工作空间中已有的变量。事实上，函数文件在执行的时候会开辟自己的工作空间，例如在 M 文件中设置断点，令程序进入调试状态后，会发现 MATLAB 界面工作空间窗口上的工具条中 stack 由 base 变成了正在调试的函数名，如果在这个函数中调用了另一个函数，stack 会再次发生变化。另外，当进入调试状态后，在命令行中建立的变量也保存在当前函数的工作空间中。进一步说，如果在函数中调用了一个脚本文件，它所用的工作空间也是这个函数空间的②。

函数文件还有下面一些特性：函数声明行之后的连续的注释行会在“help 该函数文件名”时显示出来③；函数在执行到最后一行或者遇到 end 语句时终止；函数文件中可以有多个函数，除了第一个外，其他被称为子函数或者局部函数；子函数在外部是不可见的；调用该文件名时执行的是第一个函数。

和所有程序语言一样，在顺序执行的基础上，MATLAB 还提供分支和循环控制语句。

分支语句包括 if-elseif-else-end 和 switch-case-otherwise-end，其中 if 和 elseif 后可跟随逻辑变量或数值变量（非 0 表示真，0 表示假），也可以是符号变量（要求能够转化为数值变量）；switch 后跟随变量名，case 后跟随常量，这些变量或者常量同样可以是数值、逻辑或者符号变（常）量。

循环语句包括 for-end 和 while-end，其中与 C 语言类似，可以用 continue 提前进入下一次迭代或者用 break 跳出循环体。while 后内容和 if 相同，不必解释。最复杂的是 for 语句，一般使用方法是

```
for 循环变量名 = 起始值:增量:结束值
```

事实上，MATLAB 只要求“=”后面跟一个矢量，而各元素间不必等间隔，例如

```
for n = [1,2,3,8]
```

即可实现次数为 4 的循环，其中 n 依次取值为 1,2,3 和 8。

最后，MATLAB 还支持 try-catch-end 语句捕获错误事件。这个并不常用，请读者参考帮助自学。

---

① 其实 functionname 并不一定要和文件名相同，7.1 版的 MATLAB 根据文件名而不是文件第一行声明的 functionname 调用该文件，但是为了保证和老版本兼容，同时为了避免编程中的混淆，强烈建议两者相同。

② 可以如此评价它们：霸道的函数和随遇而安的脚本。

③ 在命令窗口中输入“help 脚本文件名”，也将返回该脚本文件中从第一行开始的连续的注释文字。

第六节 阶跃信号与冲激信号

阶跃函数和冲激函数是两个特殊的数学函数，前者在零点无定义[①]，后者在零点取值无穷大。但强大的MATLAB分别定义了heaviside和dirac实现这两个函数。下面分别列出这两个函数文件的全部内容，并就函数的定义和特殊数值等相关知识进行说明。

在命令窗口中输入open heaviside，将在编辑/调试窗口中打开heaviside文件。如下所示：

```
function Y = heaviside(X)
%HEAVISIDE Step function.
%HEAVISIDE(X) is 0 for X<0,1 for X>0,and NaN for X == 0.
%HEAVISIDE(X) is not a function in the strict sense.
%See also DIRAC.

%Copyright 1993—2003 The MathWorks,Inc.
% $Revision: 1.1.6.2 $  $Date: 2004/04/16 22:23:24 $

Y = zeros(size(X));
Y(X>0) = 1;
Y(X == 0) = NaN;
```

文件第一行的首字function声明了这个文件定义的是一个函数，函数名为heaviside，输入参数X，返回值Y。文件从第二行往下连续的若干带注释的行是这个函数的说明，也就是读者在命令窗口中输入help heaviside看到的内容(注意空行后的将不会被看到)。最后三行是有实质意义的命令，分别解释如下：

```
Y = zeros(size(X));          %定义和X结构相同的全零变量
Y(X>0) = 1;                  %定义对应于X大于0的Y为1
Y(X == 0) = NaN;             %X等于零时Y非数值
```

同理可以输入open dirac，其中有实质意义的两行是

```
Y = zeros(size(X));          %定义和X结构相同的全零变量
Y(X == 0) = Inf;             %X等于零时Y无穷大
```

NaN(非数值)和Inf(无穷大)是MATLAB引入的两个重要常量，它使得很

① 主教材上册14页，未定义或定义成1/2。

多没有意义的数学运算得以顺利进行，从而保证了程序不被频繁中断。表1.3还列出了一些重要的常量和特殊变量(包括前面已经用到的常量pi)。

表1.3 重要的MATLAB常量和特殊变量

名 称	说 明	名 称	说 明
ans	默认保存最新结果的变量名	pi	圆周率π
eps	浮点数的相对误差	inf	无穷大
NaN,nan	非数	i,j	虚数单位
realmin,realmax	最小/最大浮点数	bitmax	最大正整数

第七节 常用MATLAB命令

本章已经用到了一些简单的MATLAB命令或函数，它们都在三个最基本的帮助主题中，分别是general、elmat和elfun，请读者务必调用帮助文件，浏览这三个主题中的所有命令和函数[①]。表1.4中列出了若干最常用的命令。

表1.4 MATLAB常用命令

命 令	功 能	命 令	功 能
cd	显示或改变工作目录	clc	清除命令窗口
clear	清除内存变量	clf	清楚图形窗口
copyfile	复制文件	delete	删除文件或图形对象
demo	运行实例程序	dir,ls	显示当前目录下文件
disp	显示变量内容	echo	命令窗口信息显示开关
load	载入文件中的数据	movefile	移动文件
open	打开文件供编辑	pack	整理内存碎片
pwd	显示当前工作路径	save	保存变量到文件中
type	显示文件内容	who	显示当前内存中变量

如前所述，这些命令都可以用函数形式实现，如save('filename','var1','vat2')，从而使编程变得非常方便。

① 即在命令窗口中分别输入help general，help elmat和help elfun。

第二章　MATLAB 绘图

数据可视化是 MATLAB 提供的重要功能，本章先介绍基本的绘图函数和方法，然后讲解面向对象的句柄图形，最后介绍交互式绘图方法。

第一节　基本绘图操作

一般绘图的第一步是用 figure 函数生成一个新图框[①]，然后所有的绘图指令都将在这个图框内进行。画线函数 plot 是最基本、最重要的绘图函数，它的基本调用格式是 plot(x,y,s)，其中矢量 x 和 y 分别表示线条上抽样点的两个坐标值，字符串 s 是表示颜色、线型和点型的字符组合，表 2.1 中给出了所有可能的字符及其意义。

表 2.1　plot 函数绘图类型

字　符	类　型	意　义
b/g/r/c/m/y/k	颜色	蓝/绿/红/蓝绿/紫红/黄/黑
./x/+/h/*/s/d/v/^/</>/p/o	点型	点/x/+/六角星/星号/方形/菱形/下三角/上三角/左三角/右三角/正五边形/圆圈
-/:/-./-	线型	实线/点线/点画线/虚线

如果对同类数据进行比较，经常需要在同一个坐标图中绘制多个线条，这时应调用 hold on 命令[②]声明“保持模式”，即绘制出的第二个线条不会把前面的线

① 如果不调用该函数，后面的绘图函数将针对最近操作过的图框进行；如果当前没有图框，MATLAB 会自动调用 figure 函数生成新图框。

② 或者说调用 hold('on') 函数。

条抹掉，同理 hold off 命令则起到相反的作用[①]。plot 函数也支持在一个坐标系中同时绘制多个线条，调用格式为 plot(x1,y1,s1,x2,y2,s2,…)[②]。

另外一种对同类数据进行比较的方法就是在同一个图框中对齐地放置多个坐标轴，在每个坐标轴中分别绘制一个或多个线条。这时需要调用 subplot(M,N,n)函数，其中 M 和 N 表示将原有图框分成纵向包括 M 个子图，横向包括 N 个子图的子图“矩阵”，n 指明接下来要在哪个子图中绘图，n=1 代表第一行第一列(左上角)的子图，n=2 代表第一行第二列的子图，依次类推，按照先遍历列再遍历行的顺序一直到右下角的最后一个子图。

为绘出完整的坐标图，可以用 xlabel 和 ylabel 添加横轴和纵轴的说明文字，用 title 添加整个图框或者其中某个子图的标题。最后，还可以调用 legend 函数增加图例，即按顺序标注已绘出的每个线条的名称。

例 1.1 中生成了多种信号，下面观察它们的波形特征。

例 2.1 绘制例 1.1 中各种信号的波形。

解 根据题意，以及充分展示上述基本绘图函数，将生成两个图框，在第一个图框中以不同颜色和线型绘制 x、z1、z2 和 z3；第二个图框分成左右两个子图，分别绘制 z4 和 z5，同时用虚线绘出 x 和 y，以考察它们和 z4、z5 的关系。

```
figure;                                  %生成新图框
hold on;                                 %进入绘图保持模式
plot(t,x);                               %以默认格式绘制 t-x
plot(t,z1,'r');                          %用红色实线绘制 t-z1
plot(t,z2,'k-o');                        %用黑色实线带圆圈标记绘制 t-z2
plot(t3,z3,'g--');                       %用绿色虚线绘制 t3-z3
xlabel('t(seconds)');                    %填写 X 轴说明
ylabel('signals');                       %填写 Y 轴说明
legend('x','z_1','z_2','z_3');           %填写图例
figure;                                  %生成新图框
subplot(1,2,1);                          %生成并列放置两个子图中的左边一个
plot(t,z4,'k',t,x,'b:',t,y,'r:');        %用黑实线绘制 z4，蓝和红点线绘制 x
                                         %和 y
title('z_4(t)');                         %填写标题
```

① 对于新建立的图框，默认处于 hold off 的状态。

② 事实上，plot 函数有多种调用方法及简化格式，请读者务必执行 help plot 命令仔细学习。

```
subplot(1,2,2);                    %生成并列放置两个子图中的右边一个
plot(t,z5,'k',t,x,'b:',t,y,'r:'); %同理绘制 z5
title('z_4(t)');
```

程序运行结果如图 2.1 和图 2.2 所示。

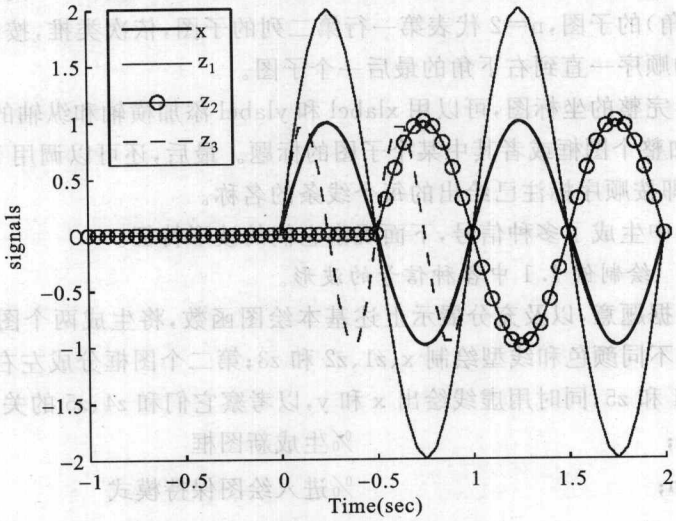


图 2.1 例 2.1 绘第一幅图

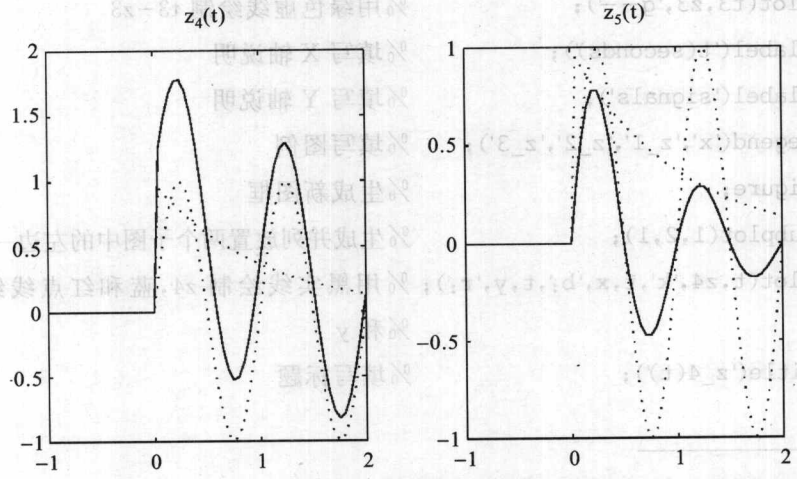


图 2.2 例 2.1 绘第二幅图

对应于用 plot 绘制连续时间信号，可以用 stem 绘制离散时间信号，请读者结合帮助自己练习。

若要绘制符号函数的波形，一种方法是利用 subs 函数计算出该函数在指定抽样时刻的抽样值，然后用 plot 函数绘制；除此方法之外，MATLAB 还提供了一个功能强大的函数 ezplot 实现符号函数的绘图功能，如下例。

例 2.2 用 ezplot 函数绘制例 1.2 中符号表达式 z5 在 t∈[−1,2]区间的波形。

解 最简单的，用一句指令就可以完成上述要求，即

```
ezplot(z5,[-1,2]);          %绘制 t 从-1 到 2π 区间的 z5(t)
```

正如它的名字一样，ezplot 函数可以便捷地绘制出符号函数的波形[①]，但它的缺点是不够灵活，比如不能指定线型和颜色等。

第二节 句柄图形

MATLAB 的数据可视化建立在对象的基础上，即一幅图的每个组成部分（如坐标轴、线、注释文本等）都是对象，可以通过对象的唯一标示句柄（Handle）访问其属性，从而实现改变各个组成部分视觉效果的目的。

任何绘图语句产生的图形效果都是对象。这些对象以层次结构组织，如图 2.3 所示。核心（Core）对象包括坐标轴（Axes）、图像（Image）、光源（Light）、线条（Line）、填充的多边形（Patch）、矩形（Rectangle）、曲面（Surface）和文本（Text）[②]。下面以坐标轴（Axes）为例介绍对象的属性及其访问方法。

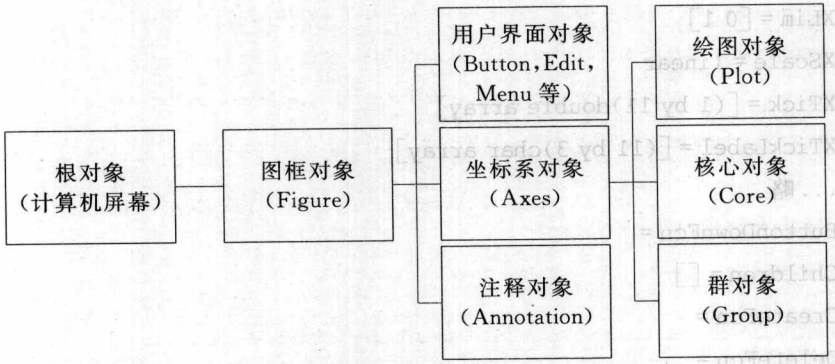


图 2.3 句柄图形对象层次结构

① ezplot 也可以绘制自定义的函数，只要定义好函数的输入输出关系即可，具体请参考 help ezplot 的详细讲解。

② 括号中的英文即是各个对象的名称，也是创建该对象的 MATLAB 命令。

句柄是对象的唯一标示，也是访问该对象的根据，熟悉C++语言的读者可以把句柄理解为指针或者引用。有两种方式获取对象的句柄，第一是在建立该对象时保留其返回值，第二是调用句柄访问函数获得当前活动对象[①]的句柄，如以下命令

```
    hf = figure                    %生成新图框，返回句柄
hf =                               %回显图框句柄 hf 的值
    2
    plot(randn(2))                 %随机绘制线条，MATLAB 将自动生成坐标轴
    ha = gca                       %获取当前坐标轴的句柄
ha =                               %回显坐标轴句柄 ha 的值
    309.0018
```

和用 gca[②] 获取当前坐标轴句柄类似，可以调用 gcf 获取当前图框句柄，调用 gco 获取当前对象句柄，还有 gcbf 和 gcbo 等，请读者查看帮助自行学习。

得到句柄之后，可以用通用函数[③] get 和 set 访问句柄的属性，以坐标轴对象为例，在命令窗口输入 get(gca)，MATLAB 回显如下：

```
get(gca)
...略...
FontName = Helvetica
FontSize = [10]
...略...
XGrid = off
XLim = [0 1]
XScale = linear
XTick = [(1 by 11)double array]
XTickLabel = [(11 by 3)char array]
...略...
ButtonDownFcn =
Children = []
CreateFcn =
DeleteFcn =
```

① 活动对象是指正在对其进行操作的对象。

② Get Current Axes 的意思。

③ 因为这两个函数不仅可以访问句柄图形，还可以访问其他任意对象的属性，因而称其为通用函数。第十六章第一节将介绍这两个函数访问 LTI 模型对象。

```
...略...
```

坐标轴共有103种属性，为节省篇幅只列出了一部分。属性FontName＝Helvetica和FontSize＝[10]表示坐标轴使用Helvetica的10pt字体显示刻度，XGrid＝off表示X轴方向是否有虚线格，XLim＝[0 1]表示X轴的起止范围是0到1，其他属性含义也都非常明显，无需解释。较难理解的是ButtonDownFcn和CreateFcn等属性，它们是坐标轴作为图形用户界面的控件时的回调函数接口，将在第十九章第五节中详细讲解。

如果要修改某个属性的值可使用set命令，例如要把X轴起止范围改为0到100，并且绘制虚线网格，可以输入

```
set(gca,'XLim',[0 100],'XGrid','on')
```

请读者查看屏幕上窗口的坐标轴是否发生了变化。如果只想获取某个属性，比如验证刚才的修改X坐标轴起止范围是否有效，可以输入

```
get(gca,'XLim')
```

将看到回显

```
ans =
     0 100
```

前面以坐标轴对象为例介绍了句柄图形的访问方法，对其他对象同样有效，为节省篇幅不再赘述，请读者结合帮助自行学习。

第三节 交互式绘图

上一节介绍了以编程方式访问句柄图形对象属性的方法，此外，MATLAB还提供了交互式的绘图方法。请读者新建一个图框，然后依次选中View菜单下的Figure Toolbar、Camera Toolbar、Plot Edit Toolbar、Figure Palette、Plot Browser和Property Editor，将看到如图2.4所示的完整的图框窗口，在其中可以手工创建图形、绘制变量并修改各种图形对象的属性。下面依次介绍三个辅助工具子窗口的功能。

首先看左侧的Figure Palette子窗口。它包括可以折叠或展开的三个板块，其中New Subplots板块用于添加子图和调整子图位置；Variables板块显示着工作空间中的变量，在某个变量上点击鼠标右键后可以从菜单上选择各种方式绘制该变量；Annotations板块列出了各种注释工具，选中某个工具后即可回到图形窗口用鼠标绘制添加注释。

右侧是Plot Browser子窗口，其中依次列出了各个子图中的所有对象，用户可以选中某个对象编辑其属性，也可以修改勾中状态隐藏和显示该对象。先选中坐标轴(Axes)对象后，再点击子窗口右下角的Add Data...按钮可以手工

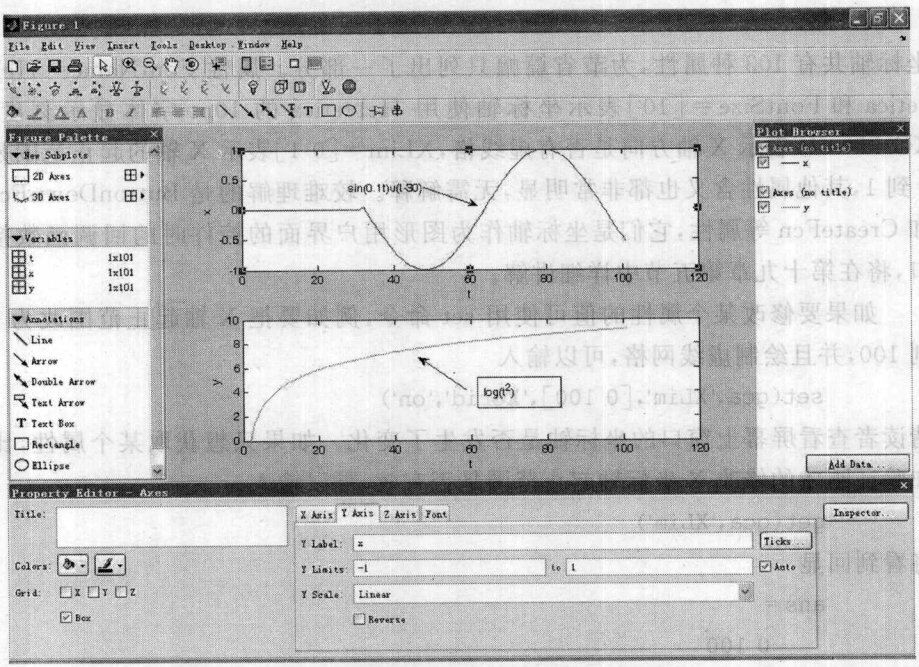


图 2.4 打开各种图形辅助工具后的图框窗口

添加各种新的绘图对象。

最下方的 Property Editor－Axes 子窗口是最常用的交互式绘图工具，它显示当前激活的图形对象的基本属性，用户可以方便直观地进行修改。若要查看该对象的全部属性，可以点击该子窗口右侧的 Inspector... 按钮，在新弹出的 Property Inspector 窗口中进行修改。

MATLBA 通过这些辅助工具子窗口为交互式绘图提供良好的支持。任何用编程实现的绘图功能都可以用鼠标键盘手工完成。编程绘图的优点是可重复性好，一劳永逸；交互式绘图的优点是直观快捷，无需记住繁杂的属性名称。两者各有优劣，请读者在使用过程中根据需要选择最合适的方式①。

除交互式绘图外，MATLAB 的图形窗口还提供了强大的输出功能，支持多种图形格式的存储和转换。绘图结束后，如果读者需要把图片插入到 Word 等文本编辑器中，可在绘图窗口中选择 Edit→Copy Figure，然后在 Word 中“粘

① 很长时间以来渴望有这样一种操作模式：用户使用交互式绘图，MATLAB 记住用户的每一步操作并保存为程序代码，以后用户若有类似绘图任务，只要略微修改代码即可完成，从而同时保持了直观快捷和可重复性两大优点。但似乎现在 MATLAB 还没有提供类似的功能。

贴”即可[①];如果需要把图片保存成文件,可点击 File→Save As 再保存为需要的格式,MATLAB 提供了包括 TIFF、JPG 和 EPS 在内的十多种图像格式[②]。

① 请复制之前勾中 Edit→Copy Options 中的 Transparent background,这样粘贴后视觉效果更好。

② 为节约篇幅,以后各章的例题中不再列出绘图的源程序,而代之以调用一个文件。读者可以从中国高校电工电子课程网中找到该文件并学习相关命令。

即"即可"①；如果需要把图片保存成文件，可点击 File→Save As 将保存为需要的格式，MATLAB 提供了包括 TIFF、JPG 和 EPS 在内的十余种图像格式②。

① 背景颜色指的是 Edit→Copy Options 中的 Transparent background，选择粘贴后背景就变为透明。

② 为节约篇幅，以后各章的例题中不再列出绘图的菜单操作，每次只说明一个文件，读者可以从中国科技电子电子课程网中获取文件参考相关命令。

第二篇

连续时间信号与系统

第二篇

连续时间信号与系统

第三章　连续时间系统的时域分析

本章首先介绍两种描述线性时不变(Linear Time-Invariant,LTI)系统的方法:输入-输出法(端口描述法)和状态方程法,然后分别讲解零状态和零输入情况下LTI系统的仿真步骤,即常系数线性常微分方程的求解方法。鉴于冲激响应和阶跃响应的重要性,第四节专门介绍这两种响应的计算方法;最后一节研究如何用数值方法实现连续卷积积分,为下一章用数值方法实现傅里叶交换做好准备工作。

本章包括四个MATLAB知识点。① 介绍多项式运算的相关函数;② 介绍一般微分方程(组)的求解方法;③ 讨论MATLAB函数返回值的问题;④ 介绍连续积分的数值计算方法。

学习重点: ① 用lsim函数仿真LTI系统;② 用离散的数值方法近似连续积分。

本章正文中首次出现的MATLAB函数及其功能

函　数	功　能	函　数	功　能
tf	建立传递函数模型	ss	建立状态方程模型
roots	多项式求解	lsim	线性系统仿真
step	计算阶跃响应	impulse	计算冲激响应
conv	计算卷积和		

第一节　引言

连续时间系统的研究方法包括输入-输出法(端口描述法)和系统状态变量分析法。首先介绍输入-输出法。“信号与系统”研究的大部分是LTI系统,可以用一元高阶微分方程描述。设激励信号$e(t)$,系统响应$r(t)$,则系统表示为

$$C_0\frac{\mathrm{d}^n}{\mathrm{d}t^n}r(t)+C_1\frac{\mathrm{d}^{n-1}}{\mathrm{d}t^{n-1}}r(t)+\cdots+C_{n-1}\frac{\mathrm{d}}{\mathrm{d}t}r(t)+C_nr(t)$$

$$=E_0\frac{\mathrm{d}^m}{\mathrm{d}t^m}e(t)+E_1\frac{\mathrm{d}^{m-1}}{\mathrm{d}t^{m-1}}e(t)+\cdots+E_{m-1}\frac{\mathrm{d}}{\mathrm{d}t}e(t)+E_me(t)$$

在 MATLAB 中，上述系统可用 tf 函数[①]建立，使用方法为 sys＝tf(b,a)，其中 a,b 即 $\boldsymbol{a}=[C_0,C_1,\cdots,C_{n-1},C_n]$和 $\boldsymbol{b}=[E_0,E_1,\cdots,E_{m-1},E_m]$，是两个行矢量，分别由系统响应和激励信号的各阶导数项的系数由高至低排列而成；返回值 sys 即为上述 LTI 系统的模型。

例 3.1(主教材例 2-3 修改) 描述如下系统

$$\frac{\mathrm{d}^3}{\mathrm{d}t^3}r(t)+7\frac{\mathrm{d}^2}{\mathrm{d}t^2}r(t)+16\frac{\mathrm{d}}{\mathrm{d}t}r(t)+12r(t)=e(t)$$

解 在命令窗口中输入

```
a = [1,7,16,12];          %定义微分方程左侧系数的行矢量 a,注意未出
                          %现的项要用零填补
b = [1];                  %定义微分方程右侧系数的行矢量 b
sys = tf(b,a);            %生成系统描述 sys
sys                       %打印显示 sys

Transfer function:
        1
---------------
s^3 + 7 s^2 + 16 s + 12
```

可见 MATLAB 以传递函数(系统函数)$H(s)$的形式描述 LTI 系统[②]。

系统状态变量分析法在主教材第十二章中有详细讲解，因为在第三节计算零输入和零状态响应时必须用到状态变量的概念，这里先作一简单介绍。前述系统是一个一元高阶微分方程，必然可以化成两个多元一阶微分方程组，其中一个描述系统状态在激励信号作用下的变化，称为状态方程

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\lambda}(t)=\boldsymbol{A}\boldsymbol{\lambda}(t)+\boldsymbol{B}\boldsymbol{e}(t)$$

另一个描述输出信号和系统状态及激励信号的关系，称为输出方程(观测方程)

① 传递函数 Transfer function 的简写。

② 传递函数的概念在主教材第四章第六节中讲授，现在读者可以利用“电路”课上学到的算子理论理解传递函数中的符号 s。

$$\boldsymbol{r}(t)=\boldsymbol{C}\boldsymbol{\lambda}(t)+\boldsymbol{D}\boldsymbol{e}(t)$$

其中 $\boldsymbol{\lambda}(t)=[\lambda_1,\lambda_2,\cdots,\lambda_k]^{\mathrm{T}}$，$\boldsymbol{e}(t)=[e_1,e_2,\cdots,e_m]^{\mathrm{T}}$ 和 $\boldsymbol{r}(t)=[r_1,r_2,\cdots,r_r]^{\mathrm{T}}$ 分别表示状态矢量、激励矢量和输出矢量，$\boldsymbol{A}_{k\times k}$，$\boldsymbol{B}_{k\times m}$，$\boldsymbol{C}_{r\times k}$ 和 $\boldsymbol{D}_{r\times m}$ 分别表示四个系数矩阵。

对于用状态方程和输出方程描述的系统模型，在 MATLAB 中用 ss 函数[①]建立。使用方法为 sys=ss(a,b,c,d)，其中四个输入参数即为上述四个矩阵，返回值 sys 表示该系统模型。

例 3.2(主教材例 12-1 修改)　描述如下系统

状态方程

$$\begin{bmatrix}\dot{\lambda}_1\\ \dot{\lambda}_2\\ \dot{\lambda}_3\end{bmatrix}=\begin{bmatrix}-2 & 0 & -1\\ 0 & -3 & 3\\ 2 & -2 & 0\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & -3\\ 0 & 0\end{bmatrix}\begin{bmatrix}e_1(t)\\ e_2(t)\end{bmatrix}$$

输出方程

$$r(t)=\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix}+\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}e_1(t)\\ e_2(t)\end{bmatrix}$$

解　在命令窗口中输入

```
A=[-2,0,-1;0,-3,3;2,-2,0];        %定义 A,B,C,D 四个矩阵
B=[1,0;0,-3;0,0];
C=[0,1,0];
D=[0,1];
sys=ss(A,B,C,D);                  %建立系统描述 sys
sys                               %打印显示 sys
```

```
a =
      x1   x2   x3
   x1 -2    0   -1
   x2  0   -3    3
   x3  2   -2    0
b =
      u1   u2
   x1  1    0
   x2  0   -3
   x3  0    0
```

① 即状态空间 state space 的简写。

```
c =      x1   x2   x3
   y1    0    1    0
d =      u1   u2
   y1    0    1
Continuous-time model.
```

上述各个矩阵中每列的上方和每行的左侧各有一个字符串，分别以 x、u 或 y 和行序号或列序号构成，其中 x 表示状态变量，u 表示激励信号，y 表示响应输出。各矩阵元素的值表示该列（上方）的变量对该行（左侧）的变量的影响程度，如 b 矩阵第一行第二列的元素反映的就是第二个激励信号对第一个状态变量的影响。

本节介绍了两种建立 LTI 系统模型的方法，在后面第十六章第一节中将进一步讲解另两种方法，并对 LTI 模型做详细解释。

第二节 微分方程式的建立与求解

微分方程的解包括齐次解和特解两部分。齐次解即系统特征方程的根，用 roots 函数计算。使用方法 a＝roots(p)，其中 p 为特征方程的系数由高至低排列构成的行矢量，返回值 a 是特征根组成的列矢量。

例 3.3(主教材例 2-3) 求微分方程

$$\frac{\mathrm{d}^3}{\mathrm{d}t^3}r(t)+7\frac{\mathrm{d}^2}{\mathrm{d}t^2}r(t)+16\frac{\mathrm{d}}{\mathrm{d}t}r(t)+12r(t)=e(t)$$

的齐次解。

解 系统特征方程为

$$a^3+7a^2+16a+12=0$$

在命令窗口中输入

```
p=[1 7 16 12];          %定义多项式 p
a=roots(p);             %求解多项式 p=0 的根 a
a                       %打印显示 a
```

```
a = -3.0000
    -2.0000 + 0.0000i
    -2.0000 - 0.0000i
```

注意有一个二重根-2[①]，从而写出齐次解为

$$r_h(t)=(A_1t+A_2)e^{-2t}+A_3e^{-3t}$$

特解即系统函数(微分方程)在给定激励信号作用下的输出，MATLAB 提供 lsim 函数[②]对 LTI 系统进行仿真。使用方法为 y=lsim(sys,u,t)，其中 sys 表示 LTI 系统，矢量 u 和 t 分别表示激励信号的抽样值和抽样时间，返回值 y 为对应于上述抽样时间的系统响应值。

例 3.4(主教材例 2-4)　给定微分方程式

$$\frac{d^2r(t)}{dt^2}+2\frac{dr(t)}{dt}+3r(t)=\frac{de(t)}{dt}+e(t)$$

如果已知：① $e(t)=t^2$；② $e(t)=e^t$，分别求两种情况下此方程的特解。

解　首先建立系统模型 sys，然后生成 10 s 的抽样时间 t 及两种输入信号 e1 和 e2，再分别激励 sys 得到响应信号 r1 和 r2，最后在两个子图上分别绘制两种输入信号及其响应。

§§§ MATLAB 文件 ex_3_4.m

```
a = [1,2,3];
b = [1,1];
sys = tf(b,a);                %建立系统描述 sys
t = [0:0.1:10]';              %定义仿真时间为 0 s 到 10 s,抽样间隔为 0.1 s
e1 = t.^2;                    %定义激励信号 e1
r1 = lsim(sys,e1,t);          %用 e1 激励 sys,输出为 r1
e2 = exp(t);                  %定义激励信号 e2
r2 = lsim(sys,e2,t);          %用 e2 激励 sys,输出为 r2
ex_3_4_plot();                %调用函数绘制输出
```

绘制结果如图 3.1 所示。注意 MATLAB 用数值方法对连续时间系统进行仿真，所以抽样间隔直接影响仿真结果的准确性。读者可以将上述脚本中抽样间隔改为 1 s，再运行程序，对比两次绘图结果有何不同。一般来说，随着抽样率的提高，仿真结果和理论值会越来越相近。

① 虚部+0.0000i 和−0.0000i 的差异由数值计算的误差导致。

② 即 LTI models simulation 的简写。

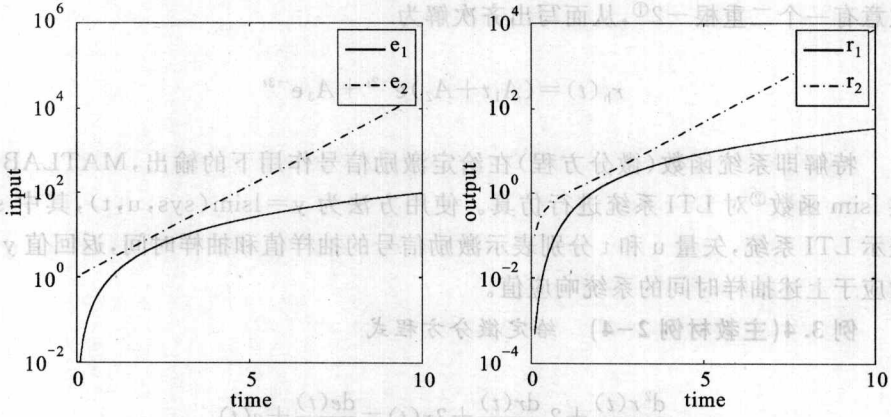


图 3.1 例 3.4 的输入和输出信号波形

MATLAB 知识点(1)——多项式

MATLAB 提供了一些处理多项式的专用函数，使用它们可以方便地求解多项式的根，对多项式进行四则运算、积分和微分等。首先，对于多项式 $p(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$，MATLAB 约定用 p 即行矢量 $\boldsymbol{p}=[a_0,a_1,\cdots,a_n]$来表示，这样多项式计算问题即转换为矢量计算问题。

poly(A)函数用于建立多项式：如果参数 A 是一个矢量，则返回以 A 的每个元素为根、最高次幂系数为 1 的多项式(矢量)；如果 A 是一个方阵，则返回矩阵 A 的特征多项式，即 $\det(\lambda\boldsymbol{I}-\boldsymbol{A})$。

若欲求多项式的值，即将 x_0 代入 $p(x)$求解 $p(x_0)$，可以用 polyval(p,x0)或者 polyvalm(p,x0)，前者以矩阵元素为计算单位(即将 x0 中的每个元素分别代入 p 的返回值再组成矩阵)，后者以矩阵为计算单位(因而 x0 必须为方阵)。求多项式的根就用 roots(p)函数，正文里已有介绍。

多项式的加法和减法可以直接对矢量进行，需要注意阶数必须相同，否则就要对低阶多项式前面做补零处理。乘法用 conv 实现，这个函数就是本章第五节将要介绍的卷积运算。除法用 deconv 实现，对应于解卷积，在主教材 7.7 节和本书第七章第四节讲授。求导和积分分别用 polyder 和 polyint 实现。这两个函数略显复杂，请读者自行练习。

第三节　零输入响应与零状态响应

在 MATLAB 中，lsim 函数还可以对带有非零起始状态的 LTI 系统进行仿真，使用方法为 y=lsim(sys,u,t,x0)，其中 sys 表示 LTI 系统，矢量 u 和 t 分别表示激励信号的抽样值和抽样时间，矢量 x0 表示该系统的初始状态，返回值 y 是系统响应值。如果只有起始状态而没有激励信号，或者令激励信号为零，则得到零输入响应。如果既有初始状态，也有激励信号，则得到完全响应。

请注意 lsim 函数只能对用状态方程描述的 LTI 系统仿真非零起始状态响应(对传递函数描述的 LTI 系统将失效)[①]。

例 3.5(主教材例 2-8)　给定图 3.2 所示电路，$t<0$ 开关 S 处于 1 的位置而且已经达到稳态，将其看做起始状态，当 $t=0$ 时，S 由 1 转向 2。分别求 $t>0$ 时 $i(t)$ 的零输入响应和零状态响应。

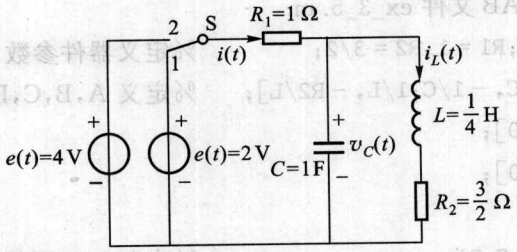


图 3.2　例 3.5 电路

解　由所示电路写出回路方程和结点方程

$$R_1 i(t)+v_C(t)=e(t)$$

$$v_C(t)=L\frac{\mathrm{d}}{\mathrm{d}t}i_L(t)+i_L(t)R_2$$

$$i(t)=C\frac{\mathrm{d}}{\mathrm{d}t}v_C(t)+i_L(t)$$

选择 $v_C(t)$ 和 $i_L(t)$ 作为状态变量，$i(t)$ 作为输出变量，得到状态方程

① MATLAB 的这种约束表明了系统状态变量分析法的重要性，也是本章开始即介绍状态方程的原因。

$$\begin{bmatrix}\dot{v}_C(t)\\ \dot{i}_L(t)\end{bmatrix}=\begin{bmatrix}-\dfrac{1}{R_1C} & -\dfrac{1}{C}\\ \dfrac{1}{L} & -\dfrac{R_2}{L}\end{bmatrix}\begin{bmatrix}v_C(t)\\ i_L(t)\end{bmatrix}+\begin{bmatrix}\dfrac{1}{R_1C}\\ 0\end{bmatrix}e(t)$$

输出方程

$$i(t)=\begin{bmatrix}-\dfrac{1}{R_1} & 0\end{bmatrix}\begin{bmatrix}v_C(t)\\ i_L(t)\end{bmatrix}+\frac{1}{R_1}e(t)$$

下面将用两种方法计算完全响应。第一种方法：首先仿真 2 V 电压 en 作用足够长时间(10 s)后系统进入稳态，从而得到稳态值 x0，再以该值作为初始值仿真 4 V 电压 e 作用下的输出 rf，即是系统的完全响应。为充分掌握 lsim 函数的使用方法，还仿真了系统的零状态响应 rzs 和零输入响应 rzi。第二种方法：构造一个激励信号，先保持 2 V 足够长时间再跳变为 4 V，然后即可以零初始状态一次仿真得到系统的完全响应 r1。

§§§ MATLAB 文件 ex_3_5.m

```
C = 1;L = 1/4;R1 = 1;R2 = 3/2;                %定义器件参数
A = [ - 1/R1/C, - 1/C;1/L, - R2/L];           %定义 A,B,C,D 四个矩阵
B = [1/R1/C;0];
C = [ - 1/R1,0];
D = [1/R1];
sys = ss(A,B,C,D);                            %建立 LTI 系统 sys
tn = [ - 10:0.01: - 0.01]';                   %生成-10 s 到-0.01 s 的抽样时
                                              %间,间隔为 0.01 s
en = 2 * (tn<0);                              %生成激励信号的抽样值,e(t)=2
[rn tn xn] = lsim(sys,en,tn);                 %仿真 t<0 时的输出信号
x0 = xn(length(en),:);                        %x0 记录了初始状态,即 0_状态
                                              %的值
t = [0:0.01:10]';                             %生成从 0 s 到 10 s 的抽样时间,
                                              %间隔为 0.01 s
e = 4 * (t> = 0);                             %生成激励信号的抽样值,e(t)=4
ezi = 0 * (t> = 0);                           %生成零输入信号的抽样值,e(t)=0
rzs = lsim(sys,e,t);                          %仿真零状态响应
rzi = lsim(sys,ezi,t,x0);                     %仿真零输入响应
rf = lsim(sys,e,t,x0);                        %仿真完全响应
r1 = lsim(sys,[en;e],[tn;t]);                 %用另一种方法直接仿真完全响应
```

```
ex_3_5_plot();                                    %绘制输出图形
```

两种方法的仿真结果如图 3.3 所示。

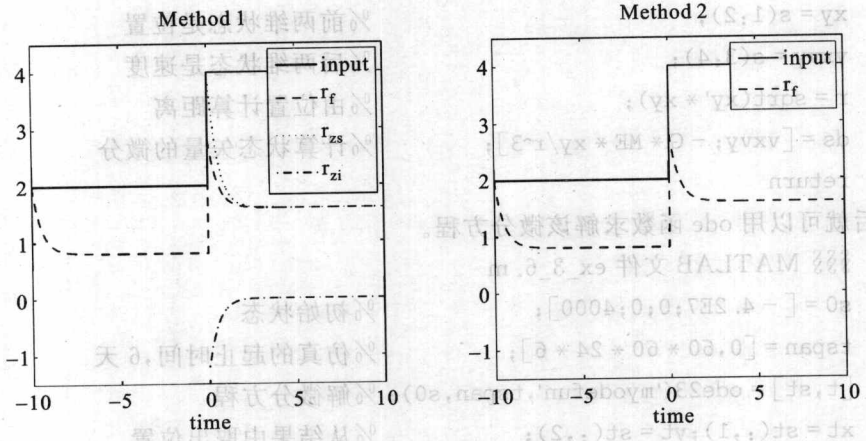


图 3.3 例 3.5 运行结果

MATLAB 知识点(2)——解微分方程(组)

本节正文讲授的方法只能用于求解常系数微分方程，对于一般的非常系数微分方程，MATLAB 提供了更一般的方法——ode 类函数求解，下面以“信号与系统”课程之外的一个实际例子给出说明。

例 3.6 绘制地球卫星的运行轨道。以卫星轨道为平面，地球位置为坐标原点，定义卫星的二维运动状态 $\boldsymbol{s}=[x,y,v_x,v_y]^{\mathrm{T}}$，其中 x,y 表示位置，v_x,v_y 表示速度，可知运动方程为

$$\dot{\boldsymbol{s}}=\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{v}_x\\ \dot{v}_y\end{bmatrix}=\begin{bmatrix}v_x\\ v_y\\ -GM_{\mathrm{E}}\dfrac{x}{(x^2+y^2)^{\frac{3}{2}}}\\ -GM_{\mathrm{E}}\dfrac{y}{(x^2+y^2)^{\frac{3}{2}}}\end{bmatrix}$$

其中引力常数 $G=6.672\times10^{-11}\ \mathrm{m^3/(kg\cdot s^2)}$，地球质量 $M_{\mathrm{E}}=5.97\times10^{24}\ \mathrm{kg}$，假设初值 $x(0)=-4.2\times10^{7}\ \mathrm{m}$，$y(0)=0\ \mathrm{m}$，$v_x(0)=0\ \mathrm{m/s}$，$v_y(0)=4\times10^{3}\ \mathrm{m/s}$。

解 首先根据公式写出描述该微分方程的函数 myodefun. m。

```
%%% MATLAB 文件 myodefun.m
function ds = myodefun(t,s)          %输入时间和状态矢量,输出微分
G = 6.672E-11;ME = 5.97E24;
xy = s(1:2);                         %前两维状态是位置
vxvy = s(3:4);                       %后两维状态是速度
r = sqrt(xy' * xy);                  %由位置计算距离
ds = [vxvy; -G * ME * xy/r^3];       %计算状态矢量的微分
return
```

然后就可以用 ode 函数求解该微分方程。

```
%%% MATLAB 文件 ex_3_6.m
s0 = [-4.2E7;0;0;4000];              %初始状态
tspan = [0,60 * 60 * 24 * 6];        %仿真的起止时间,6 天
[t,st] = ode23('myodefun',tspan,s0); %解微分方程
xt = st(:,1);yt = st(:,2);           %从结果中取出位置
figure,hold on,box on;
plot(xt,yt,'k');                     %绘制卫星轨道
[XE,YE,ZE] = sphere(10);             %生成单位球面数据
RE = 6.4E6;                          %地球半径
mesh(RE * XE,RE * YE,0 * ZE);        %在平面上绘制地球
axis('image');                       %调整横纵坐标为相同比例
```

程序运行结果如图 3.4 所示。MATLAB 提供了包括 ode23,ode45 和 ode113 等多个求解微分方程的函数,这些函数主要是采用龙格-库塔法求解,ode 后面的数字即代表不同的阶数,如 ode23 表示使用普通 2-3 阶法,而 ode15s 则表示变阶法,因而各种方法的适用范围、计算精度和复杂度各不相同,请读者在使用前仔细阅读帮助。

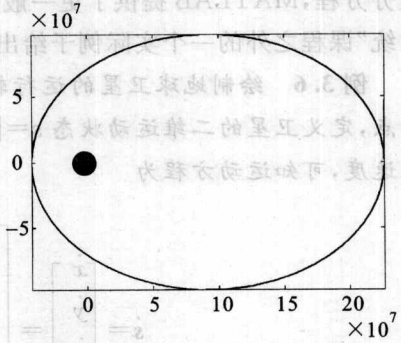


图 3.4 例 3.6 地球和卫星轨道

第四节 冲激响应与阶跃响应

如果分别用冲激信号和阶跃信号作激励,lsim 函数可仿真出冲激响应和阶

跃响应。但鉴于这两种响应的重要性，为简化操作，MATLAB 专门提供了 impulse(sys)和 step(sys)两个函数分别直接产生 LTI 系统的冲激响应和阶跃响应，其中 sys 表示 LTI 系统模型。

例 3.7(主教材例 2-9 和例 2-10)　对图 3.2 所示电路，分别求电流 $i(t)$对激励 $e(t)=\delta(t)$和 $e(t)=u(t)$的冲激响应 $h(t)$和阶跃响应 $g(t)$。

解　先求得电路的微分方程表示

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}i(t)+7\frac{\mathrm{d}}{\mathrm{d}t}i(t)+10i(t)=\frac{\mathrm{d}^2}{\mathrm{d}t^2}e(t)+6\frac{\mathrm{d}}{\mathrm{d}t}e(t)+4e(t)$$

参考主教材，由理论方法可推导出系统的冲激响应 $h(t)$和阶跃响应 $g(t)$为

$$h(t)=\delta(t)+\left(-\frac{4}{3}\mathrm{e}^{-2t}+\frac{1}{3}\mathrm{e}^{-5t}\right)u(t)$$

$$g(t)=\left(\frac{2}{3}\mathrm{e}^{-2t}-\frac{1}{15}\mathrm{e}^{-5t}+\frac{2}{5}\right)u(t)$$

下面演示 MATLAB 求解冲激响应和阶跃响应的两种方法，以及 lsim 函数的多种调用方式。本着由简入繁的原则，首先绘制阶跃响应，然后再绘制冲激响应。

§§§ MATLAB 文件 ex_3_6.m

```
a = [1,7,10];
b = [1,6,4];
sys = tf(b,a);                    %定义 LTI 系统模型
t = [0:0.01:3];                   %生成 0 到 3 s,间隔 0.01 s 的抽样时
                                  %间

figure;                           %生成新图框,将在其中绘制四个子图
subplot(2,2,1);                   %在左上角的子图中绘制阶跃响应
step(sys);                        %用 step 函数仿真 sys 的阶跃响应并
                                  %绘图

subplot(2,2,2);                   %在右上角的子图中用第二种方法绘
                                  %制阶跃响应
x_step = zeros(size(t));          %根据定义构造阶跃信号 x_step
x_step(t>0) = 1;
x_step(t == 0) = 1/2;             %lsim 函数不支持输入 NaN,所以 x_
                                  %step(0)=1/2
lsim(sys,x_step,t);               %仿真 x_step 激励 sys 的响应并绘图
```

```
subplot(2,2,3);                    %在左下角的子图中绘制冲激响应
[h1,t1] = impulse(sys,t);          %仿真 sys 的冲激响应并保存在 h1
                                   %中,不绘图
plot(t1,h1,'k');                   %用黑色实线绘制冲激响应
title('Impulse Response');         %设置子图标题
xlabel('Time(sec)');               %设置横坐标说明
ylabel('Amplitude');               %设置纵坐标说明

subplot(2,2,4);                    %在右下角的子图中用第二种方法绘
                                   %制冲激响应
x_delta = zeros(size(t));          %根据定义构造冲激响应 x_delta
x_delta(t == 0) = 100;             %lsim 函数不支持输入 Inf,所以令 x_
                                   %delta(0) = 100,因为抽样间隔是
                                   %0.01,选择100 可保证数值积分为 1

[y1,t] = lsim(sys,x_delta,t);      %仿真 x_delta 激励 sys 的响应并保存
                                   %在 y1 中,不绘图
y2 = y1 - x_delta;                 %从响应中减去一个冲激信号得到 y2
plot(t,y2,'k');                    %用黑色实线绘制 y2
title('Impulse Response');         %设置标题和横纵坐标说明
xlabel('Time(sec)');
ylabel('Amplitude');
```

运行结果如图 3.5 所示,可见用两种方法绘制出的响应基本相同。请读者注意 impulse 函数没有绘出冲激响应中 $\delta(t)$ 分量。认真阅读 help impulse 就会发现这一点[①],因而在"自制"的冲激响应 y1 中减去了冲激信号 x_delta,从而得到和 impulse 函数基本相同的结果(所以请读者记住 y1 才是完整的冲激响应)。另外,读者一定观察到:在调用 lsim、impulse 或者 step 函数时如果加上返回值,例如 h=impulse(sys,t),则 h 将被赋予冲激响应在抽样时刻 t 的值,同时不再自动绘出波形。

① help impulse 第一段的最后一句是"For continuous systems with direct feedthrough, the infinite pulse at t=0 is disregarded.",感谢清华大学电子工程系五字班姚卯青同学指出讲义中的这个问题。

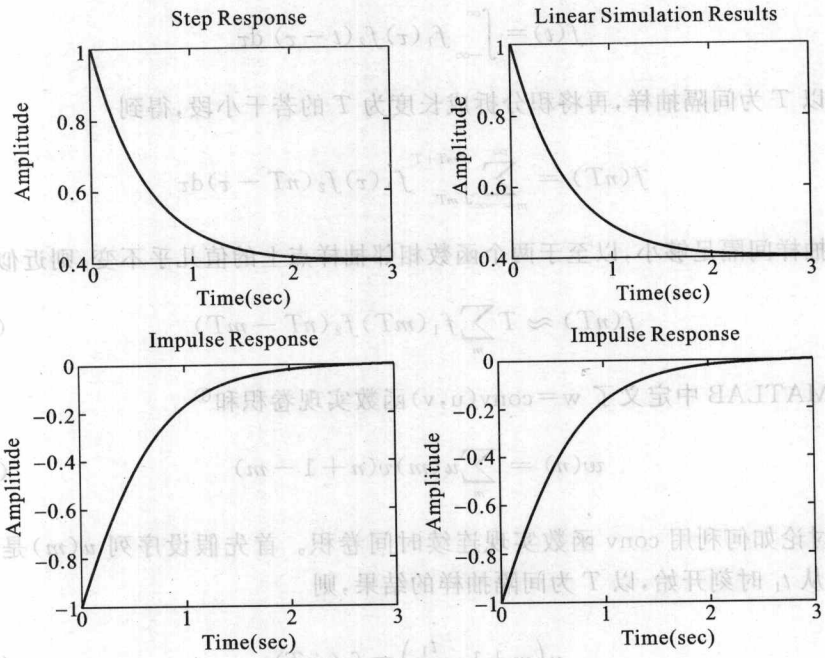


图 3.5　例 3.7 运行结果

MATLAB 知识点(3)——是否需要返回值

经过一段时间的学习，读者可能已经深刻感受到几乎每个 MATLAB 函数都存在多种调用格式，对应于不同的输入输出变量。很多函数都有这样的特点：如果不带返回值调用，MATLAB 会主动利用返回值完成一些操作，比如把返回值以图形方式表现出来，就像 impulse 一样。这样的双重功能为用户提供了很大的方便：如果只要了解返回值的特征，就让 MATLAB 画出来；如果关心某个具体的值，就让它算出来。似乎唯一的不足是：MATLAB 自动绘制的图形比较难操纵，比如无法修改标题字号或者改变线条颜色。

第五节　卷积

在 MATLAB 中，连续时间的卷积运算可以用数值方法计算近似值。首先对卷积公式

$$f(t)=\int_{-\infty}^{\infty} f_1(\tau)f_2(t-\tau)\,\mathrm{d}\tau$$

两侧以 T 为间隔抽样，再将积分拆成长度为 T 的若干小段，得到

$$f(nT)=\sum_{m=-\infty}^{\infty}\int_{mT}^{mT+T} f_1(\tau)f_2(nT-\tau)\mathrm{d}\tau$$

假设抽样间隔足够小，以至于两个函数相邻抽样点上的值几乎不变，则近似有

$$f(nT)\approx T\sum_m f_1(mT)f_2(nT-mT) \tag{3.1}$$

MATLAB 中定义了 w=conv(u,v)函数实现卷积和[①]

$$w(n)=\sum_m u(m)v(n+1-m) \tag{3.2}$$

下面讨论如何利用 conv 函数实现连续时间卷积。首先假设序列 $u(m)$ 是函数 $f_1(t)$ 从 t_1 时刻开始，以 T 为间隔抽样的结果，则

$$u\left(m+1-\frac{t_1}{T}\right)=f_1(mT) \tag{3.3}$$

假设 t_1 是 T 的整数倍。同理假设 $v(m)$ 是对函数 $f_2(t)$ 从 t_2 时刻开始，以相同间隔抽样得到的序列，即

$$v\left(m+1-\frac{t_2}{T}\right)=f_2(mT) \tag{3.4}$$

将式(3.3)和式(3.4)代入式(3.1)，有

$$f(nT)\approx T\sum_m u\left(m+1-\frac{t_1}{T}\right)v\left(n-m+1-\frac{t_2}{T}\right) \tag{3.5}$$

定义 $m'=m+1-\frac{t_1}{T}$，$n'=n+1-\frac{t_1+t_2}{T}$ 并代入式(3.5)，得到

$$f(nT)\approx T\sum_m u(m')v(n'+1-m') \tag{3.6}$$

对比式(3.2)和式(3.6)有

$$f(nT)\approx Tw(n')=Tw\left(n+1-\frac{t_1+t_2}{T}\right) \tag{3.7}$$

① 即序列的卷积，在主教材 7.6 节中讲授。

即 $w(n)$近似为从 t_1+t_2 时刻开始以 T 为间隔对 $f(t)$抽样得到的序列，从而可以用 conv 函数实现连续时间卷积。在以上推导的基础上，定义了一个函数 conv1 实现连续时间卷积[①]：

§§§ MATLAB 文件 conv1.m

```
function[w,tw] = conv1(u,tu,v,tv)
%输入参数：
%u 和 v 表示两个序列，tu 和 tv 分别表示它们的抽样时间
%返回值：
%w 和 wt 分别表示卷积结果及其抽样时间
T = tu(2) - tu(1);
w = T * conv(u,v);
tw = tu(1) + tv(1) + T * [0:length(u) + length(v) - 2]';
```

下面例题将演示卷积运算以及 conv1 函数的使用方法。

例 3.8(主教材图 2-12 修改)　已知某系统冲激响应为 $h(t)=t/2,0<t<2$。以 -0.5 s 开始，宽度为 1.5 s，幅度为 1 的矩形脉冲 $e(t)$激励该系统，求输出信号 $r(t)$。

解　§§§ MATLAB 文件 ex_3_8.m

```
t = [-1:0.01:4]';                    %生成从-1 s 到 4 s，间隔 0.01 s 的抽
                                     %样时间 t
e = (t> - 1/2&t<1);                  %定义激励信号 e
h = (t>0&t<2). * t/2;                %定义冲激响应 h
[r1,t1] = conv1(e,t,h,t);            %用卷积计算系统输出 r1，相应的抽样
                                     %时间为 t1
tr = t1(t1> = - 1&t1< = 4);          %从 t1 中选择和 t 相同起止时刻的抽
                                     %样时间 tr
r = r1(t1> = - 1&t1< = 4);           %用类似方法选择 tr 对应的输出 r
ex_3_8_plot();                       %绘制激励信号、冲激响应和输出
```

运行结果如图 3.6 所示。

① 清华大学电子工程系五字班耿泉同学建议定义此函数并提供函数原型，特此致谢。

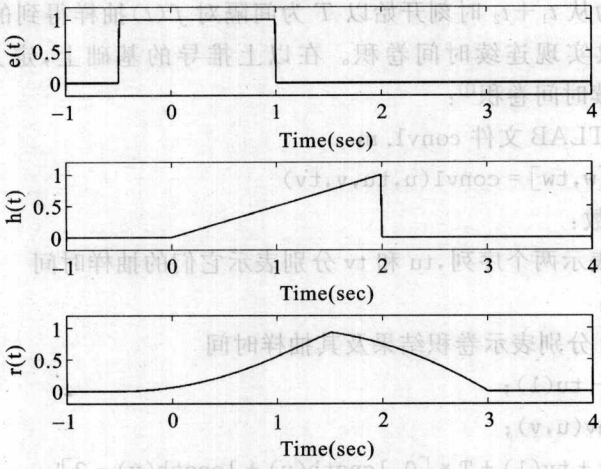


图 3.6 例 3.8 运行结果

MATLAB 知识点(4)——连续时间信号的数值计算

自然界中绝大部分物理量都是连续的，因而处理连续信号是工程应用中的一般需求。现有的数字计算机系统建立在离散时间和抽样数据的离散表示基础之上，因而无法从根本上解决连续信号处理的问题。积分是最基本的连续信号运算形式，其数值计算形式为分段求和，即

$$F(a,b)=\int_a^b f(t)\mathrm{d}t \approx \frac{b-a}{N}\sum_{n=0}^{N-1} f\left(a+n\frac{b-a}{N}\right)$$

随着分段数量 N 的增加，数值计算的结果将越来越接近真实值。请注意求和之后必须乘以分段长度 $(b-a)/N$。

高级的数值计算方法包括插值、拟合等，常用的 MATLAB 指令包括 fit 等。

第六节 小结

本章首先讲解了连续时间 LTI 系统的建立和仿真方法，重点是仿真函数 lsim 的灵活运用。该函数有多种调用方式，能够仿真零输入响应和零状态响应。虽然 lsim 函数也可以计算冲激响应和阶跃响应，但为简化编程工作，MATLAB 为这两个重要响应分别提供了专用函数 impulse 和 step。希望读者认识到，MATLAB 提供了很多专用函数完成各种常见的特定运算，命名方式即

为该运算的英文名称或其简写。请记住这些函数名以便需要时迅速调用。最后，本章讲解了卷积运算的数值实现方法，并简单讨论了数值计算的一般问题。

练习题

1.（主教材习题 2-9 修改）已知某系统微分方程为

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}r(t)+\frac{\mathrm{d}}{\mathrm{d}t}r(t)+r(t)=\frac{\mathrm{d}}{\mathrm{d}t}e(t)+e(t)$$

分别用两种方法计算其冲激响应 $h(t)$ 和阶跃响应 $g(t)$，对比理论结果进行验证。

2. 请编写一个自定义函数[F,tF]=intl(f,tf,a)实现数值积分，其中 f 和 tf 分别用列矢量表示待积函数的抽样值和抽样时间，a 表示积分的起始时间，F 和 tF 分别表示积分结果的抽样值和抽样时间。请设计一个积分运算验证 intl 的计算结果是否正确。

3. 请用 intl 函数对练习题 1 系统的冲激响应进行积分，考察积分结果和阶跃响应的关系。

第四章　傅里叶变换

和主教材讲解顺序不同，本章首先介绍傅里叶变换的符号运算方法和数值实现方法，再讲解傅里叶级数的数值实现方法，最后验证卷积定理。先讲变换再讲级数的顺序更有助于增进对概念的理解。本章结合傅里叶变换的数值实现，对比循环、矢量和矩阵三种计算思路讲解提高 MATLAB 运行效率的关键问题。本章还定义了函数 prefourier，用于完成傅里叶变换数值计算方法的准备工作，该函数将在后续章节多次使用。

本章包括三个 MATLAB 知识点。① 程序优化技巧；② 生成连续周期信号的方法；③ 自定义函数中的参数有效性检查。

学习重点： ① 傅里叶变换的数值实现方法；② 程序优化方法。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
fourier	傅里叶变换	ifourier	傅里叶逆变换
kron	张量积(叉乘)	sqrt	平方根

第一节　傅里叶变换

MATLAB 提供了符号函数 fourier 和 ifourier 实现傅里叶变换和逆变换，见下例。

例 4.1　计算 $tu(t)$ 和 $\sin t$ 的傅里叶变换。

解　在命令窗口中输入

```
syms t                              %定义符号 t
F1 = fourier(t * heaviside(t))      %计算 tu(t)的傅里叶变换 F1
F1 =
```

```
i * (pi * dirac(1,w) * w^2 + i)/w^2
F2 = fourier(sin(t));                    %计算 sin(t)的傅里叶变换 F2
F2 =
i * pi * (dirac(w + 1) - dirac(w - 1))
```

即 $F_1=\mathrm{i}\dfrac{\pi\delta'(\omega)\omega^2+\mathrm{i}}{\omega^2}=\mathrm{i}\pi\delta'(\omega)-\dfrac{1}{\omega^2}$，$F_2=\mathrm{i}\pi[\delta(\omega+1)-\delta(\omega-1)]$。注意 dirac(1,w)表示 $\delta'(\omega)$。

工程应用中经常需要对抽样数据进行傅里叶分析，这种情况下往往无法得到信号的解析表达式，因而数值计算方法是应用傅里叶变换的主要途径。下面将重点介绍计算傅里叶变换和逆变化的数值方法。傅里叶变换的表达式为

$$F(\omega)=\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t=\int_{t_1}^{t_2}f(t)\mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t \tag{4.1}$$

其中$[t_1,t_2]$为 $f(t)$的主要取值区间，定义 $T=t_2-t_1$ 为区间长度。在该区间内抽样 N 个点，抽样间隔为 $\Delta t=\dfrac{T}{N}$，得到

$$F(\omega)=\frac{T}{N}\sum_{n=0}^{N-1}f(t_1+n\Delta t)\mathrm{e}^{-\mathrm{j}\omega(t_1+n\Delta t)}$$

用上式可以算出任意频点的傅里叶变换值。假设 $F(\omega)$的主要取值区间位于$[\omega_1,\omega_2]$，要计算其间均匀抽样的 K 个值，则有

$$F(\omega_1+k\Delta\omega)=\frac{T}{N}\sum_{n=0}^{N-1}f(t_1+n\Delta t)\mathrm{e}^{-\mathrm{j}(\omega_1+k\Delta\omega)(t_1+n\Delta t)} \tag{4.2}$$

其中 $\Delta\omega=\dfrac{\Omega}{K}$为频域抽样间隔，$\Omega=\omega_2-\omega_1$ 为带宽。

下面利用同样的方法研究傅里叶逆变换，由于 $f(t)$的频带主要位于$[\omega_1,\omega_2]$，近似有

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\,\mathrm{d}\omega=\frac{\Omega}{2\pi K}\sum_{k=0}^{K-1}F(\omega_1+k\Delta\omega)\mathrm{e}^{\mathrm{j}(\omega_1+k\Delta\omega)t} \tag{4.3}$$

仍然只关心时域抽样点上的值，得到

$$f(t_1+n\Delta t)=\frac{\Omega}{2\pi K}\sum_{k=0}^{K-1}F(\omega_1+k\Delta\omega)\mathrm{e}^{\mathrm{j}(\omega_1+k\Delta\omega)(t_1+n\Delta t)} \tag{4.4}$$

现在即可用式(4.2)计算傅里叶变换的 K 个频域抽样值，进而可以用式(4.4)恢复时域信号。通过下例具体讲解实现该公式的三种方法，同时介绍用矩阵计算代替循环加速 MATLAB 运算的技巧。

例 4.2(主教材图 3-21 修改) 请绘制矩形脉冲

$$f(t)=\begin{cases}1 & |t|<\dfrac{1}{2}\\ 0 & \text{其他}\end{cases}$$

的波形($t\in[-1,1]$)和频谱 $F(\omega)$($\omega\in[-8\pi,8\pi]$),并利用你计算得到的频谱恢复时域信号 $f_s(t)$,比较和原信号 $f(t)$的差别。

解 直接计算法(方法一)

直接编程实现式(4.2)和式(4.4),对傅里叶变换和逆变换分别用二重循环实现。

§§§MATLAB 文件 ex_4_2_1.m

```
T = 2;                                  %定义时域抽样区间长
                                        %度
N = 200;                                %定义时域抽样点数
t = linspace( - T/2,T/2 - T/N,N)';      %定义时域抽样点
f = 0 * t;                              %初始化时域信号
f(t > - 1/2&t<1/2) = 1;                 %为时域信号赋值
OMG = 16 * pi;                          %定义频域抽样区间长
                                        %度
K = 100;                                %定义频域抽样点数
omg = linspace( - OMG/2,OMG/2 - OMG/K,K)';  %定义频域抽样点
F = 0 * omg;                            %初始化频谱
for k = 1:K                             %循环计算每个频域抽
                                        %样点的频谱
for n = 1:N                             %用循环实现式(4.2)
                                        %中的求和运算
    F(k) = F(k) + T/N * f(n) * exp( - j * omg(k) * t(n));
  end
end
fs = 0 * t;                             %初始化合成信号
for n = 1:N                             %循环计算每个时域抽
                                        %样点的合成信号
  for k = 1:K                           %用循环方式实现式
                                        %(4.4)中的求和运算
    fs(n) = fs(n) + OMG/2/pi/K * F(k) * exp(j * omg(k) * t(n));
  end
```

```
end
ex_4_2_plot();                                    %绘制输出图形
```

结果如图 4.1 所示。由于恢复时域信号时丢弃了高频分量，因而合成信号和原始信号相比，在跳变处有较大的波动，这就是 Gibbs 现象。

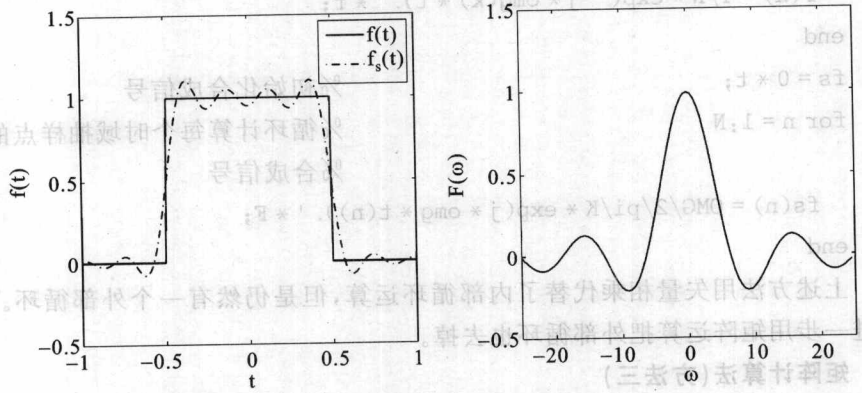


图 4.1　例 4.2 的波形和频谱

上述方法的优点是原理直观，编程简便，缺点是计算量大，耗时较多。由于 MATLAB 对矢量和矩阵运算做了很大的优化，所以如果可以将式(4.2)和式(4.4)中的循环改为矢量运算必将极大地降低计算复杂度。

矢量计算法(方法二)

式(4.2)中的累加和可以用内积实现，即一个行矢量右乘一个列矢量，将已知时域抽样值 $f(t_1)$，$f(t_1+\Delta t)$，…，$f(t_2-\Delta t)$写成列矢量，则式(4.2)改写为

$$F(\omega_1+k\Delta\omega)=\frac{T}{N}\left[\mathrm{e}^{-\mathrm{j}(\omega_1+k\Delta\omega)t_1}\quad \mathrm{e}^{-\mathrm{j}(\omega_1+k\Delta\omega)(t_1+\Delta t)}\quad \cdots\quad \mathrm{e}^{-\mathrm{j}(\omega_1+k\Delta\omega)(t_2-\Delta t)}\right]\begin{bmatrix} f(t_1)\\ f(t_1+\Delta t)\\ \vdots\\ f(t_2-\Delta t)\end{bmatrix} \tag{4.5}$$

同样，式(4.4)改写为

$$f(t_1+n\Delta t)=\frac{\Omega}{2\pi K}\left[\mathrm{e}^{\mathrm{j}\omega_1(t_1+n\Delta t)}\quad \mathrm{e}^{\mathrm{j}(\omega_1+\Delta\omega)(t_1+n\Delta t)}\quad \cdots\quad \mathrm{e}^{\mathrm{j}(\omega_2-\Delta\omega)(t_1+n\Delta t)}\right]\begin{bmatrix} F(\omega_1)\\ F(\omega_1+\Delta\omega)\\ \vdots\\ F(\omega_2-\Delta\omega)\end{bmatrix} \tag{4.6}$$

§§§MATLAB 文件 ex_4_2_2.m[①]

```
F = 0 * omg;                          %初始化频谱
for k = 1:K                           %循环计算每个频域抽样点的
                                      %频谱
  F(k) = T/N * exp( - j * omg(k) * t). ' * f;
end
fs = 0 * t;                           %初始化合成信号
for n = 1:N                           %循环计算每个时域抽样点的
                                      %合成信号
  fs(n) = OMG/2/pi/K * exp(j * omg * t(n)). ' * F;
end
```

上述方法用矢量相乘代替了内部循环运算,但是仍然有一个外部循环。下面进一步用矩阵运算把外部循环也去掉。

矩阵计算法(方法三)

由式(4.5)和式(4.6)可见,要在时域和频域上的两个矢量之间互求,因而可用矩阵形式表述为

$$\begin{bmatrix} F(\omega_1) \\ F(\omega_1+\Delta\omega) \\ \vdots \\ F(\omega_2-\Delta\omega) \end{bmatrix} = \frac{T}{N} \begin{bmatrix} e^{-j\omega_1 t_1} & e^{-j\omega_1(t_1+\Delta t)} & \cdots & e^{-j\omega_1(t_2-\Delta t)} \\ e^{-j(\omega_1+\Delta\omega)t_1} & e^{-j(\omega_1+\Delta\omega)(t_1+\Delta t)} & \cdots & e^{-j(\omega_1+\Delta\omega)(t_2-\Delta t)} \\ \vdots & \vdots & & \vdots \\ e^{-j(\omega_2-\Delta\omega)t_1} & e^{-j(\omega_2-\Delta\omega)(t_1+\Delta t)} & \cdots & e^{-j(\omega_2-\Delta\omega)(t_2-\Delta t)} \end{bmatrix} \begin{bmatrix} f(t_1) \\ f(t_1+\Delta t) \\ \vdots \\ f(f_2-\Delta t) \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} f(t_1) \\ f(t_1+\Delta t) \\ \vdots \\ f(t_2-\Delta t) \end{bmatrix} = \frac{\Omega}{2\pi K} \begin{bmatrix} e^{j\omega_1 t_1} & e^{j(\omega_1+\Delta\omega)t_1} & \cdots & e^{j(\omega_2-\Delta\omega)t_1} \\ e^{j\omega_1(t_1+\Delta t)} & e^{j(\omega_1+\Delta\omega)(t_1+\Delta t)} & \cdots & e^{j(\omega_2-\Delta\omega)(t_1+\Delta t)} \\ \vdots & \vdots & & \vdots \\ e^{j\omega_1(t_2-\Delta t)} & e^{j(\omega_1+\Delta\omega)(t_2-\Delta t)} & \cdots & e^{j(\omega_2-\Delta\omega)(t_2-\Delta t)} \end{bmatrix} \begin{bmatrix} F(\omega_1) \\ F(\omega_1+\Delta\omega) \\ \vdots \\ F(\omega_2-\Delta\omega) \end{bmatrix} \tag{4.8}$$

简写为

$$\boldsymbol{F} = \frac{T}{N}\boldsymbol{U}\boldsymbol{f} \tag{4.9}$$

$$\boldsymbol{f} = \frac{\Omega}{2\pi K}\boldsymbol{V}\boldsymbol{F} \tag{4.10}$$

① 为节约篇幅,初始化和绘图等与 ex_4_2_1.m 文件中重复的代码不再列出。下同。

其中 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 分别代表式(4.7)和式(4.8)中的变换矩阵。

现在一个新的问题出现了：如何有效地构造矩阵 $\boldsymbol{U}$,$\boldsymbol{V}$？以 $\boldsymbol{U}$ 矩阵为例，各个元素变化的部分在指数项(不包括－j)上，仔细观察可发现所有指数项构成的矩阵表现为两个矢量的张量积(叉乘)，即有

$$\begin{bmatrix}\omega_1\\ \omega_1+\Delta\omega\\ \vdots\\ \omega_2-\Delta\omega\end{bmatrix}\otimes\begin{bmatrix}t_1 & t_1+\Delta t & \cdots & t_2-\Delta t\end{bmatrix}=\begin{bmatrix}\omega_1 t_1 & \omega_1(t_1+\Delta t) & \cdots & \omega_1(t_2-\Delta t)\\ (\omega_1+\Delta\omega)t_1 & (\omega_1+\Delta\omega)(t_1+\Delta t) & \cdots & (\omega_1+\Delta\omega)(t_2-\Delta t)\\ \vdots & \vdots & & \vdots\\ (\omega_2-\Delta\omega)t_1 & (\omega_2-\Delta\omega)(t_1+\Delta t) & \cdots & (\omega_2-\Delta\omega)(t_2-\Delta t)\end{bmatrix}$$

MATLAB 提供了 kron 函数[①]实现张量积[②]。利用该函数，只用四行程序即可实现傅里叶变换和逆变换。

§§§ MATLAB 文件 ex_4_2_3.m

```
U = exp( - j * kron(omg,t. '));   %定义变换矩阵
F = T/N * U * f;                  %左乘矩阵实现傅里叶变换
V = exp(j * kron(t,omg. '));      %定义逆变换矩阵
fs = OMG/2/pi/K * V * F;          %左乘逆变换矩阵实现傅里叶逆变换
```

前面给出了实现傅里叶变换和逆变换的三种方法，其中最后一种矩阵计算方法在讲离散系统时还会有介绍，届时可以用更简单的快速傅里叶变换 fft 函数实现。

在本科高年级或者研究生课程中选修“泛函分析”等课程，会讲到线性算子等于矩阵算子，即线性变换都可以用矩阵形式表征。在以后编程时一定要牢记上述结论，首先把待实现算法写成矩阵形式，然后再用 MATLAB 直接实现，切忌使用循环等程序控制级的编程方式！

MATLAB 知识点(5)——程序优化技巧

MATLAB 基于逐条解释命令的方法执行(尽管 7.0 版也支持编译出独立于开

① Kronecker tensor product 的简写。

② 为了简化操作，当在命令窗口中输入列矢量乘以行矢量时，MATLAB 将其理解为叉乘，即上述公式在 MATLAB 中既可以写成 kron(omg,t.')，也可以直接写成 omg * t.'。注意只有这种情况下乘法 * 才等同于叉乘，其他情况下 MATLAB 会对矩阵大小不匹配的乘法报错。

发环境的可执行文件，但事实上很少用到），因而对循环和判断等此类需要多条命令语句组合实现的代码的运行效率非常低。与此相对应的是，MATLAB 对矩阵运算做了大量的优化，很多矩阵相关的函数都用低级语言编写并针对处理器特性进行了优化，因而运行效率很高。所以优化 MATLAB 代码的首要原则就是尽量少使用程序控制命令，尽量多用矩阵和矢量操作实现。对下述循环

```
for n = 1:1e6
  sin(n);
end
```

MATLAB 需要运行大约 2 s 才能重现提示符进入准备状态（处理器 Intel P IV 2.4 G，内存 512 M，下同），而

```
sin([1:1e6]);
```

几乎是一瞬间即可执行结束。

由于对代码进行解释执行，MATLAB 不会对变量预先分配内存，因而也不要求程序中对变量进行预先定义，这种灵活的方式方便了用户编程，但同时带来潜在的低效率问题。例如下面代码（注意先用 clear 释放所有内存变量）

```
clear all
x = zeros(1e6,1);
for n = 1:1e6
  x(n) = sin(n);
end
```

同样在 2 s 后返回，而

```
clear all
for n = 1:1e6
  x(n) = sin(n);
end
```

则要运行至少 1 min[①]。后一段程序由于没有对 x 做初始化，每次都要为 x 重新分配内存，因而效率极低。

矩阵化和变量预定义是提高 MATLAB 代码效率的主要手段，读者一定要熟练掌握。但同时需要注意，MATLAB 对矩阵的优化只做到了二维，因而巧妙地把算法设计成三维矩阵实现也是意义不大的。最后再强调程序优化的必要性：尽管 MATLAB 支持用 Ctrl+C 中断当前程序的运行，但在实际运用中往往

① 我没有耐心等它运行完，运行 1 min 后我中断程序的时候循环刚刚执行了 3 万多次，而每次循环时间肯定要比上一次的长。

不能有效执行[①]!

第二节　周期信号的傅里叶级数分析

周期为 T_1 的连续信号 $f(t)$的指数形式傅里叶级数为

$$f(t)=\sum_{k=-\infty}^{\infty}F_k e^{jk\omega_1 t} \tag{4.11}$$

其中 $\omega_1=\frac{2\pi}{T_1}$,系数

$$F_k=\frac{1}{T_1}\int_{t_0}^{t_0+T_1}f(t)e^{-jk\omega_1 t}dt \tag{4.12}$$

上述信号的三角函数形式的傅里叶级数为

$$f(t)=a_0+\sum_{k=1}^{\infty}[a_k\cos(k\omega_1 t)+b_k\sin(k\omega_1 t)] \tag{4.13}$$

其中系数

$$a_0=F_0\quad\begin{cases}a_k=F_k+F_{-k}\\ b_k=j(F_k-F_{-k})\end{cases}(k=1,2,\cdots) \tag{4.14}$$

下面可以先对式(4.11)和式(4.12)做离散化有限项表示,然后再利用上一节介绍的矩阵表示方法计算傅里叶级数的指数形式的系数和展开式,最后表示出三角函数形式级数的系数。

首先假设傅里叶级数收敛,即可以用 F_{k_1},F_{k_1+1},…,F_{k_2} 共 K 项近似原函数

$$f(t)=\sum_{k=k_1}^{k_2}F_k e^{jk\omega_1 t} \tag{4.15}$$

将所关心的从 t_0 到 t_0+T 之间的 N 个抽样值

$$f(t_0+n\Delta t)=\sum_{k=k_1}^{k_2}F_k e^{jk\omega_1(t_0+n\Delta t)} \tag{4.16}$$

写成矩阵形式为

---

① 实践证明:如果希望在 MATLAB 进入忙碌的运行状态后(状态条上显示 Busy 字样)迅速中断它,任务管理器是解决此问题的最有效工具。

$$\begin{bmatrix} f(t_0) \\ f(t_0+\Delta t) \\ \vdots \\ f(t_0+T_1-\Delta t) \end{bmatrix} = \begin{bmatrix} e^{jk_1\omega_1 t_0} & e^{j(k_1+1)\omega_1 t_0} & \cdots & e^{jk_2\omega_1 t_0} \\ e^{jk_1\omega_1 (t_0+\Delta t)} & e^{j(k_1+1)\omega_1 (t_0+\Delta t)} & \cdots & e^{jk_2\omega_1 (t_0+\Delta t)} \\ \vdots & \vdots & & \vdots \\ e^{jk_1\omega_1 (t_0+T_1-\Delta t)} & e^{j(k_1+1)\omega_1 (t_0+T_1-\Delta t)} & \cdots & e^{jk_2\omega_1 (t_0+T_1-\Delta t)} \end{bmatrix} \cdot \begin{bmatrix} F_{k_1} \\ F_{k_1+1} \\ \vdots \\ F_{k_2} \end{bmatrix} \tag{4.17}$$

离散化数值近似后，式(4.12)变为

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f(t_0+n\Delta t) e^{-jk\omega_1 (t_0+n\Delta t)} \tag{4.18}$$

写成矩阵形式

$$\begin{bmatrix} F_{k_1} \\ F_{k_1+1} \\ \vdots \\ F_{k_2} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} e^{-jk_1\omega_1 t_0} & e^{-jk_1\omega_1 (t_0+\Delta t)} & \cdots & e^{-jk_1\omega_1 (t_0+T_1-\Delta t)} \\ e^{-j(k_1+1)\omega_1 t_0} & e^{-j(k_1+1)\omega_1 (t_0+\Delta t)} & \cdots & e^{-j(k_1+1)\omega_1 (t_0+T_1-\Delta t)} \\ \vdots & \vdots & & \vdots \\ e^{-jk_2\omega_1 t_0} & e^{-jk_2\omega_1 (t_0+\Delta t)} & \cdots & e^{-jk_2\omega_1 (t_0+T_1-\Delta t)} \end{bmatrix} \cdot \begin{bmatrix} f(t_0) \\ f(t_0+\Delta t) \\ \vdots \\ f(t_0+T_1-\Delta t) \end{bmatrix} \tag{4.19}$$

对比矩阵化的傅里叶级数的数值表达式和傅里叶变换的数值表达式，可以发现其形式非常相似，因而编程实现方法也大致相同。

例 4.3(主教材图 3-6 修改) 绘制周期 $T_1=1$、幅度 $E=1$ 的对称方波的前 10 项傅里叶级数的系数(三角函数形式)，并用前 5 项恢复原信号。

解 如前所述，可以先计算指数形式的系数，再转化成三角函数形式。本题中函数为偶对称，因而正弦分量全部为零，由式(4.13)可写出前 K 项(未计入直流项)级数的合成公式

$$\begin{bmatrix} f(t_0) \\ f(t_0+\Delta t) \\ \vdots \\ f(t_0+T_1-\Delta t) \end{bmatrix} = \begin{bmatrix} 1 & \cos(\omega_1 t_0) & \cdots & \cos(K\omega_1 t_0) \\ 1 & \cos(\omega_1 (t_0+\Delta t)) & \cdots & \cos(K\omega_1 (t_0+\Delta t)) \\ \vdots & \vdots & & \vdots \\ 1 & \cos(\omega_1 (t_0+T_1-\Delta t)) & \cdots & \cos(K\omega_1 (t_0+T_1-\Delta t)) \end{bmatrix} \cdot$$

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_K \end{bmatrix} \tag{4.20}$$

§§§MATLAB 文件 ex_4_3.m

```
E = 1;                                          %定义方波幅度
T1 = 1;                                         %定义方波周期
omg1 = 2 * pi/T1;                               %定义基频
N = 1000;                                       %定义时域抽样
                                                %点数
t = linspace( - T1/2,T1/2 - T1/N,N)';           %生成时域抽样点
f = 0 * t;                                      %初始化时域信号
f(:) = - E/2;
f(t> - T1/4&t<T1/4) = E/2;
k1 = - 10;                                      %确定系数的起止
                                                %下标
k2 = 10;
k = [k1:k2]';                                   %生成系数下标
                                                %序列
F = 1/N * exp( - j * kron(k * omg1,t.')) * f;   %求指数形式傅里
                                                %叶级数的系数
a0 = F(11);                                     %转换到三角函数
                                                %形式的系数
ak = F(12:21) + F(10: - 1:1);
fs = cos(kron(t,[0:5] * omg1)) * [a0;ak(1:5)];  %用前五个系数合
                                                %成原函数
ex_4_3_plot();                                  %绘制图形
```

绘图输出结果如图 4.2 所示，其中 $f_k(t)=a_k\cos(k\omega_1 t)$。

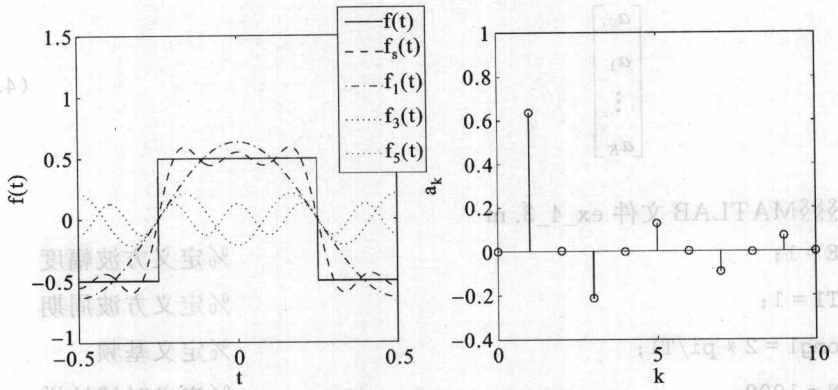


图 4.2 例 4.3 时域波形和傅里叶级数

MATLAB 知识点(6)——生成周期性连续信号

第一章详细讲解了用逻辑运算生成常用非周期连续信号的方法。若要生成周期性连续信号,单纯用逻辑运算比较困难。为了方便编程,MATLAB 提供了两个函数 square 和 sawtooth 分别用于生成最常见的矩形波和锯齿波(包括三角波)。

这两个函数的使用方法和 sin 函数相似,示例如下:

```
t = [0:0.01:6 * pi];                %抽样时间
f1 = sin(t);                        %周期 2π 的正弦信号
f2 = square(t,50);                  %周期 2π,幅度 1 的方波,等价于
                                    %square(t)
f3 = square(t,20);                  %周期 2π,幅度 1,占空比为 20%的矩
                                    %形波
f4 = 5 * square(t * 2 * pi);        %周期 1,幅度 5 的方波
f5 = sawtooth(t,1);                 %周期 2π,幅度 1 的锯齿波(升齿),等价
                                    %于 sawtooth(t)
f3 = sawtooth(t,0.5);               %周期 2π,幅度 1 的三角波
f4 = sawtooth(t * 2 * pi,0);        %周期 1,幅度 1 的锯齿波(降齿)
```

第三节 卷积特性(卷积定理)

后面章节将频繁用到傅里叶变换,为简化编程,定义函数 prefourier 完成用

矩阵左乘实现傅里叶变换前的准备工作。该函数的输入参数包括时域起止范围、时域抽样点数、频域起止范围和频域抽样点数，返回值包括时域和频域抽样点、傅里叶变换和逆变换矩阵。

§§§MATLAB 文件 prefourier.m

```
function[t,omg,FT,IFT] = prefourier(Trg,N,OMGrg,K)
T = Trg(2) - Trg(1);                                    %时域范围
t = linspace(Trg(1),Trg(2) - T/N,N)';                   %生成抽样时间点
OMG = OMGrg(2) - OMGrg(1);                              %频域范围
omg = linspace(OMGrg(1),OMGrg(2) - OMG/K,K)';
                                                        %生成抽样频率点
FT = T/N * exp( - j * kron(omg,t.'));                   %构造带有系数的傅
                                                        %里叶变换矩阵
IFT = OMG/2/pi/K * exp(j * kron(t,omg.'));              %构在带有系数的傅
                                                        %里叶逆变换矩阵
```

下面利用 prefourier 函数，用例题验证卷积定理：时域卷积对应于频域相乘。

例 4.4(主教材例 3-9)　已知

$$f(t)=\begin{cases}E\left(1-\dfrac{2|t|}{\tau}\right) & |t|\leqslant\dfrac{\tau}{2}\\ 0 & |t|>\dfrac{\tau}{2}\end{cases}$$

利用卷积定理求三角脉冲的频谱。

解　下面用两种方法计算三角脉冲的频谱：一是直接对三角脉冲做傅里叶变换；二是利用卷积定理，矩形脉冲的卷积是三角脉冲，所以可以先计算矩形脉冲的频谱，再取其平方。将两种计算结果绘制在一起，验证卷积定理的正确性。

§§§MATLAB 文件 ex_4_4.m

```
[t,omg,FT,IFT] = prefourier([ - 1,1],500,[ - 25,25],500);
tau = 1;
E = 1;
g = sqrt(2 * E/tau)(t > - tau/4&t <tau/4);    %由定义生成 g(t)的抽样点
f = E * (1 - 2 * abs(t)/tau). * (t > - tau/2&t <tau/2);
                                              %由定义生成 f(t)的抽样点
G = T/N * U * g;                              %由定义计算 G(ω)
F = T/K * U * f;                              %由定义计算 F(ω)
Fe = G. * G;                                  %由卷积定理计算 Fe(ω)
```

```
fe = OMG/2/pi/K * V * Fe;                    %由 Fe(ω)逆变换得到 fe(t)
ex_4_4_plot();                               %绘制输出波形
```

计算结果如图 4.3 所示。可见利用卷积定理计算得到的三角脉冲的频谱和直接计算得到的频谱完全相同。但由该频谱逆变换得到的时域波形 $f_e(t)$ 和三角脉冲相比有些“钝化”，这正是高频分量被截断的表现。

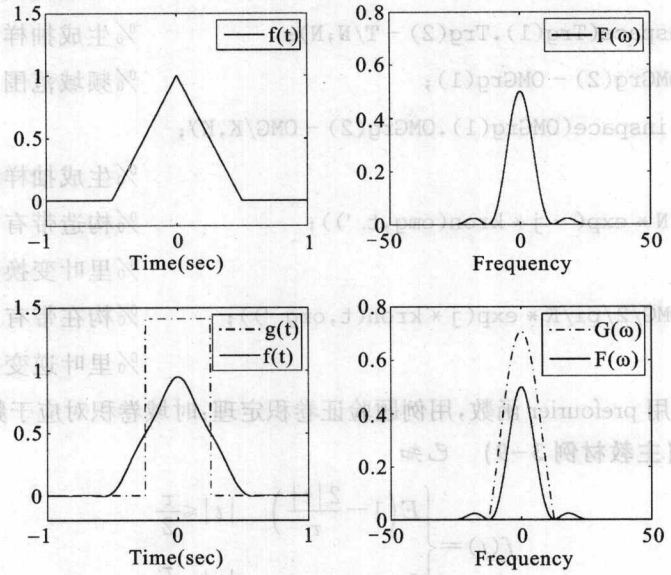


图 4.3 例 4.4 运行结果

MATLAB 知识点(7)——提高函数的稳健性

第三章第五节引入了一个自定义的卷积函数 conv1，本节又定义了傅里叶变换的预备函数 prefourier。从功能上说这两个函数已经完成，但从程序的稳健性来看，它们和 MATLAB 自带的标准函数相比仍有很大差距。请读者在命令窗口中输入 open conv 打开标准的卷积函数如下：①

```
function c = conv(a,b)
%注释部分，略去
na = length(a);                              %获取序列 a 的长度 na
nb = length(b);                              %获取序列 b 的长度 nb
```

① 这里只列出了感兴趣的代码，中文注释由本书作者添加。

```
%如果 na 不等于 a 的元素个数(即 a 是矩阵而非矢量)或者 b 也如此
if na~ = numel(a) || nb~ = numel(b)%numel(a)返回矩阵 a 中元素的个数
%显示错误信息并中断运行
  error('MATLAB:conv:AorBNotVector','A and B must be vectors.');
end
%实现卷积功能,略去
```

一个良好的程序,除了无差错、高效率地实现预定功能外,还必须判断输入参数的有效性,否则可能导致无意义的运行结果,或者因格式错误根本无法运行。另外,程序还应该向用户显示无效的参数及改正建议,供用户参考。

第四节　小结

本章主要讲解傅里叶变换的数值实现方法。由于 MATLAB 是解释性语言,并且已经针对二维矩阵运算做了大量优化,同样的算法用矩阵计算的效率要远高于用循环实现。因而推荐读者使用乘变换矩阵的方法高效地进行傅里叶变换和逆变换。为方便读者,本章自定义了 prefourier 函数完成构造变换矩阵等准备工作,并且以验证卷积定理为目标对该函数进行了测试。

练习题

1. 如图 4.4 所示锯齿波信号,分别取一个周期的抽样数据 $x_1(t)$,$0 \leqslant t < 1$ 和五个周期的数据 $x(t)$,$0 \leqslant t < 5$,计算其傅里叶变换 $X_1(\omega)$ 和 $X(\omega)$,比较有何不同并解释原因。

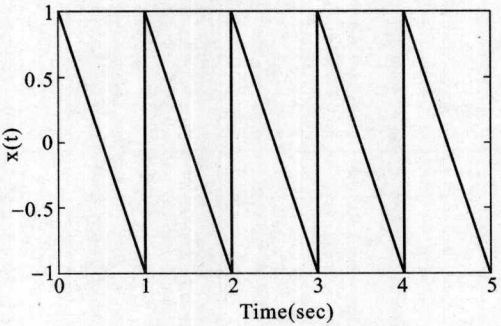


图 4.4　练习题 2 图

2. 请编写函数 F=fsana(t,f,N),计算周期信号 f 的前 N 个指数形式的傅里叶级数系数,t 表示 f 对应的抽样时间(均为一个周期);再编写函数 f=fssyn(F,t),由傅里叶级数系数

F 合成抽样时间 t 对应的函数。设计信号这两个函数是否正确。

3. 利用 fsana 和 fssyn 计算练习题 1 中 $x_1(t)$的前 10 个傅里叶级数系数 F_i，$0 \leqslant i < 10$，请用这些系数合成周期为 0.5 的锯齿波 $y(t)$，$0 \leqslant t < 2$ 并绘出图形。

第五章　拉普拉斯变换、连续时间系统的 s 域分析

本章首先介绍求解拉普拉斯变换(简称拉氏变换)和逆变换的符号方法,以及用部分分式展开求拉氏逆变换的留数方法;然后和连续时间系统时域模型相呼应,复习用传递函数描述和仿真 LTI 系统的方法;接下来从时域和频域两个角度研究系统零、极点位置的影响;最后动画演示二阶谐振系统的极点远离虚轴过程中冲激响应和频率响应的变化。

本章包括三个 MATLAB 知识点。① 符号运算函数;② 虚数单位变量;③ 结构的概念。

学习重点:① 用部分分式展开法求拉氏逆变换;② 计算连续时间系统的频响,绘制零、极点分布图。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
laplace	拉氏变换	ilaplace	拉氏逆变换
residue	部分分式展开	zero	计算系统零点
pole	计算系统极点	pzmap	绘制零、极点分布图
zp2tf	由零、极点计算传递函数	tf2zp	由传递函数计算零极点
freqs	计算频率响应	pause	暂停运行

第一节　拉普拉斯变换和逆变换

MATLAB 用符号函数 laplace 和 ilaplace 实现(单边)拉氏变换和逆变换,如下例所示。

例 5.1 计算 t^3 和 $\sin(\omega t)$的拉氏变换。

解 在命令窗口中输入

```
syms t w                          %定义符号
F1 = laplace(t^3)                 %计算拉氏变换 F1
F1 =
6/s^4
F2 = laplace(sin(w * t))          %计算拉氏变换 F2
F2 =
w/(s^2 + w^2)
```

例 5.2(主教材例 4-8) 求下示函数的拉氏逆变换

$$F(s)=\frac{10(s+2)(s+5)}{s(s+1)(s+3)}$$

解 在命令窗口中输入

```
syms s;                                   %定义符号 s
f = ilaplace(10 * (s + 2) * (s + 5)/s/(s + 1)/(s + 3))
                                          %求拉氏逆变换
```

运行结果为

```
f =
 - 10/3 * exp( - 3 * t) - 20 * exp( - t) + 100/3
```

用符号函数求拉氏变换和逆变换的方法很简单,但有较大的局限性,而且无助于增进对概念的理解。推荐在 MATLAB 协助下用部分分式展开的方法求解拉氏逆变换①。MATLAB 提供了部分分式展开的函数 residue②,下面分有实数极点且无重根、有共轭复数极点和有多重极点三种情况分别介绍该函数的使用方法。

例 5.3(主教材例 4-9) 求下示函数的拉氏逆变换。

$$F(s)=\frac{s^3+5s^2+9s+7}{(s+1)(s+2)}$$

解 本例为有实数极点且无重根。在命令窗口中输入

```
b = [1,5,9,7];          %F(s)分子多项式的系数
a1 = [1,1];             %F(s)分母多项式第一个分式的系数
a2 = [1,2];             %F(s)分母多项式第二个分式的系数
a = conv(a1,a2);        %计算 F(s)分母多项式的系数
```

① 拉氏变换主要用于求解微分方程和分析系统频域特征,因而极少对抽样数据用数值方法计算拉氏变换或逆变换,工程应用中也不需要这样做。

② 也就是计算留数,请用联机帮助查看该函数的调用方法。

```
[r,p,k] = residue(b,a)        %部分分式展开,得到系数 r,极点 p 和自由项 k
r = -1                        %两个部分分式系数
2
p = -2                        %两个极点(特征根)
-1
k = 1 2                       %自由项
```

将系数与极点配对得到 $F(s)$的部分分式展开形式

$$F(s)=s+2-\frac{1}{s+2}+\frac{2}{s+1}$$

从而直接写出拉氏逆变换式

$$f(t)=\delta'(t)+2\delta(t)-\mathrm{e}^{-2t}+2\mathrm{e}^{-t}\quad(t\geqslant 0)$$

例 5.4(主教材例 4-10)　求下示函数的拉氏逆变换。

$$F(s)=\frac{s^2+3}{(s^2+2s+5)(s+2)}$$

解　本例有共轭复数极点。在命令窗口中输入

```
b = [1,0,3];                  %F(s)分子多项式的系数
a1 = [1,2,5];                 %F(s)分母多项式第一个分式的系数
a2 = [1,2];                   %F(s)分母多项式第二个分式的系数
a = conv(a1,a2);              %计算 F(s)分母多项式的系数
[r,p,k] = residue(b,a)        %部分分式展开,得到系数 r,极点 p 和自由项 k
```

结果为

```
r =                           %三个部分分式系数
 -0.2000 + 0.4000i
 -0.2000 - 0.4000i
 1.4000
p =                           %三个极点(特征根)
 -1.0000 + 2.0000i
 -1.0000 - 2.0000i
 -2.0000
k =                           %自由项为空,因为分子阶数小于分母阶数
 []
```

将常数与极点配对得到 $F(s)$的部分分式展开形式

$$F(s)=\frac{-0.2+\mathrm{j}0.4}{s+1-\mathrm{j}2}+\frac{-0.2-\mathrm{j}0.4}{s+1+\mathrm{j}2}+\frac{1.4}{s+2}$$

由主教材公式(4-57)

$$\mathscr{L}^{-1}\left[\frac{A+\mathrm{j}B}{s+\alpha-\mathrm{j}\beta}+\frac{A-\mathrm{j}B}{s+\alpha+\mathrm{j}\beta}\right]=2\mathrm{e}^{-\alpha t}[A\cos(\beta t)-B\sin(\beta t)]$$

可直接写出拉氏逆变换式

$$f(t)=1.4\mathrm{e}^{-2t}-2\mathrm{e}^{-t}[0.2\cos(2t)+0.4\sin(2t)](t\geqslant 0)$$

例 5.5(主教材例 4-12) 求下示函数的拉氏逆变换。

$$F(s)=\frac{s-2}{s(s+1)^3}$$

解 本例有多重极点。在命令窗口中输入

```
b = [1, -2];                                  %F(s)分子多项式的系数
a1 = [1,0];                                   %F(s)分母多项式第一个分式的
                                              %系数
a2 = [1,1];                                   %F(s)分母多项式第二个分式的
                                              %系数
a = conv(conv(a1,a2),conv(a2,a2));            %计算 F(s)分母多项式的系数
[r,p,k] = residue(b,a)                        %部分分式展开
```

结果为

```
  r =                                         %四个部分分式系数
   2.0000
   2.0000
   3.0000
  -2.0000
p =                                           %四个极点(或者说一重极点、三重极点各一个)
  -1.0000
  -1.0000
  -1.0000
   0
k =                                           %自由项为空,因为分子阶数小于分母阶数
   []
```

注意有一个三重极点,将常数与极点配对得到 $F(s)$ 的部分分式展开形式

$$F(s)=\frac{2}{s+1}+\frac{2}{(s+1)^2}+\frac{3}{(s+1)^3}-\frac{2}{s}$$

从而直接写出逆变换式

$$f(t)=2e^{-t}+2te^{-t}+\frac{3}{2}t^2e^{-t}-2\quad(t\geqslant 0)$$

注意 residue 输出结果中重根对应的多项式以升幂顺序排列。

MATLAB 知识点(8)——符号运算函数

符号变量可以组合成符号多项式，多项式的合并同类项用 collect，因式分解用 factor，用 simplify 和 simple 进行化简。进行变量替换或者构造复合多项式，可以用 subs 和 compose 函数，finverse 用于求反函数。

微积分方面，用 limit 函数计算极限，diff 函数做微分和求导，jacobian 函数实现多元函数的求导，即返回雅可比矩阵；最后，用 int 进行积分运算。

MATLAB 用 linsolve 求解线性方程组，fsolve 求解非线性方程组(注意必须先将该方程组定义成函数)，solve 函数则用于求解一般的方程组。dsolve 函数用来求解微分方程组。

第二节　系统函数(网络函数)$H(s)$

回顾前面介绍的 tf 函数，可知 MATLAB 用系统函数 $H(s)$描述 LTI 系统。

例 5.6(主教材例 4-17 修改)　图 5.1 所示电路在 $t=0$ 时开关 S 闭合，接入信号源 $e(t)=\sin(3t)$，电感起始电流等于零，求电流 $i(t)$。

解　由电路可知传递函数为

$$H(s)=\frac{1}{s+1}$$

§§§MATLAB 文件 ex_5_6.m

```
sys = tf(1,[0.1,0.1]);          %定义系统传递函数 H(s)
t = [0:0.01:10]';               %定义抽样时间 t
e = sin(3 * t);                 %生成激励信号 e(t)
i = lsim(sys,e,t);              %仿真电流信号 i(t)
ex_5_6_plot( );                 %绘制激励和电流
```

输出结果如图 5.2 所示。

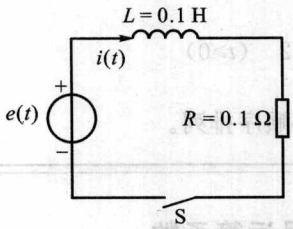


图 5.1 例 5.6 电路

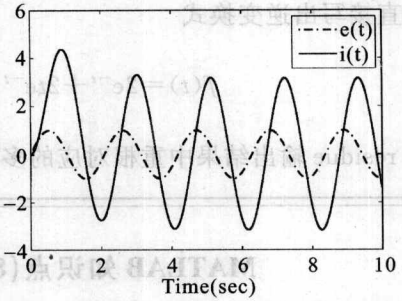


图 5.2 例 5.6 运行结果

第三节 由系统函数零、极点分布决定时域特性

计算零、极点可以用 roots 函数，若参数为传递函数 $H(s)$ 的分子多项式系数 b，则得到零点；若为分母多项式系数 a，则得到极点。MATLAB 还提供了 zero(sys)和 pole(sys)函数直接计算零、极点，其中 sys 表示系统传递函数。另外，[p,z]=pzmap(sys)函数也具有计算极点 p 和零点 z 的功能；不带返回值的 pzmap(sys)则绘制出系统的零、极点分布图。

零、极点和传递函数的多项式系数一样，可作为 LTI 系统的描述方法。MATLAB 提供了(b,a)=zp2tf(z,p,k)和[z,p,k]=tf2zp(b,a)两个函数用于在上述两种描述方法之间进行转换，其中 k 为用零、极点表示传递函数时的系统增益。

例 5.7(主教材习题 4-22 修改) 当 $F(s)$ 极点(一阶)落于图 5.3 所示 s 平面图中各方框所处位置时，画出对应的 $f(t)$ 波形填入方框中。

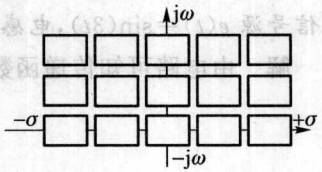


图 5.3 例 5.7 框图

解

§§§ MATLAB 文件 ex_5_7.m

```
t = [0:.1:40]';                          %定义抽样时间
figure,id = 1;                           %生成新图框,定义当前子图标号 id=1
for omega = .5:-.25:0                    %循环 ω=[0.5,0.25,0]
  for sigma = -.06:.03:.06               %循环 σ=[-0.06,-0.03,…,0.06]
    p = sigma + j * omega;               %定义极点 p=σ+jω
    if omega~ = 0                        %如果极点不在实轴上
    p = [p;p'];                          %则再添加一个共轭极点
    end
```

```
        [b,a] = zp2tf([],p,1);          %由零、极点转换为传递函数的多项式
                                        %系数
        subplot(3,5,id);                %生成标号为 id 的子图框
        impulse(b,a,t);                 %绘制抽样时间为 t 的冲激响应
        set(gca,'YLim',[-20,20]);       %设置 Y 轴范围以求最好视觉效果
        id = id + 1;                    %子图标号加 1
    end                                 %内层 σ 循环结束
end                                     %外层 ω 循环结束
```

输出结果如图 5.4 所示。可见随着极点从虚轴左侧向右移动到虚轴右侧，其冲激响应由衰减变为发散；随着极点由实轴向上、下两侧移动，冲激响应由单调变化转为振荡变化，且振荡频率逐渐增加。

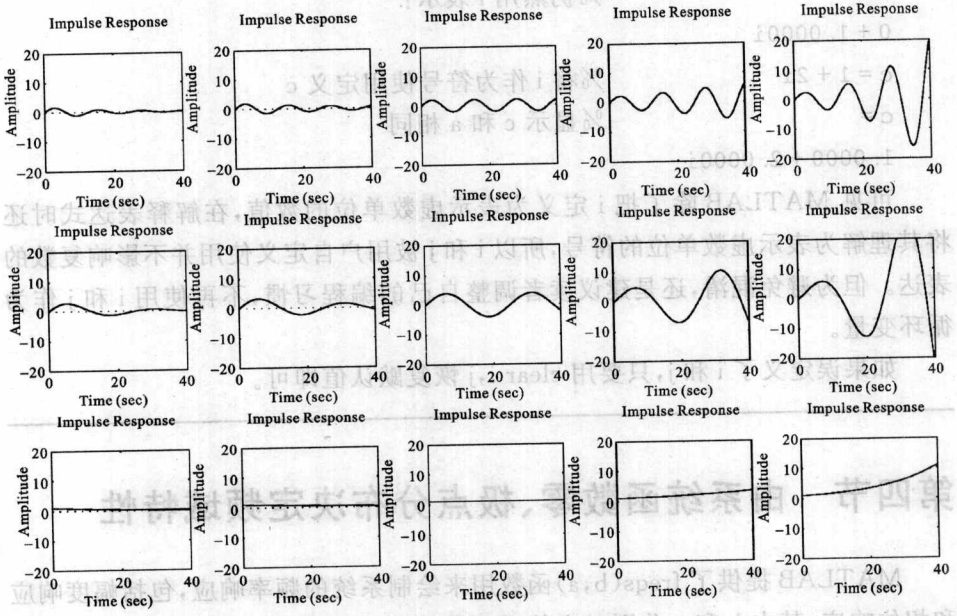


图 5.4　例 5.7 运行结果

MATLAB 知识点(9)——虚数单位 i,j 及常量恢复

和数学公式中经常以 i 和 j 作为下标一样，包括 Fortran 和 C 语言在内的大部分计算机语言教科书都习惯性地把这两个字符作为循环变量使用，但在 MATLAB 中 i 和 j 被默认定义为表示虚数单位的常量。虽然用户还可以将其

自定义成任意类型变量或者数据结构，但这样做很容易带来混乱。请看下列命令和输出。

```
clear all                    %清空所有变量，此时 i 恢复为默认的虚数单位
a = 1 + 2 * i                %定义 a
a =
1.0000 + 2.0000i
i = 1;                       %将 i 自定义为 1
b = 1 + 2 * i                %和 a 相仿定义 b
b =                          %显示 b 和 a 不同，说明 i 已被修改
3
sqrt( - 1)                   %查看现在表示虚数单位的符号
ans =                        %仍然用 i 表示！
0 + 1.0000i
c = 1 + 2i                   %将 i 作为符号使用定义 c
c =                          %显示 c 和 a 相同
1.0000 + 2.0000i
```

可见 MATLAB 除了把 i 定义为表示虚数单位的数值，在解释表达式时还将其理解为表示虚数单位的符号，所以 i 和 j 被用户自定义使用并不影响复数的表达。但为避免混淆，还是建议读者调整自己的编程习惯，不再使用 i 和 j 作为循环变量。

如果误定义了 i 和 j，只要用 clear i,j 恢复默认值即可。

第四节 由系统函数零、极点分布决定频域特性

MATLAB 提供了 freqs(b,a)函数用来绘制系统的频率响应，包括幅度响应和相位响应，其中 b 和 a 分别对应传递函数的分子和分母多项式系数。如果将调用方式改为 H＝freqs(b,a,omega)，则不绘图输出，而是计算抽样点 omega 处的频响并传递到 H 中。

例 5.8(主教材习题 4-39) 若 $H(s)$零、极点分布如图 5.5 所示，试讨论它们分别是哪种滤波网络(低通、高通、带通、带阻)。

解 §§§MATLAB 文件 ex_5_8.m

```
data = struct('title',{'(a)','(b)','(c)',...
                                    %定义 data 为结构数组
```

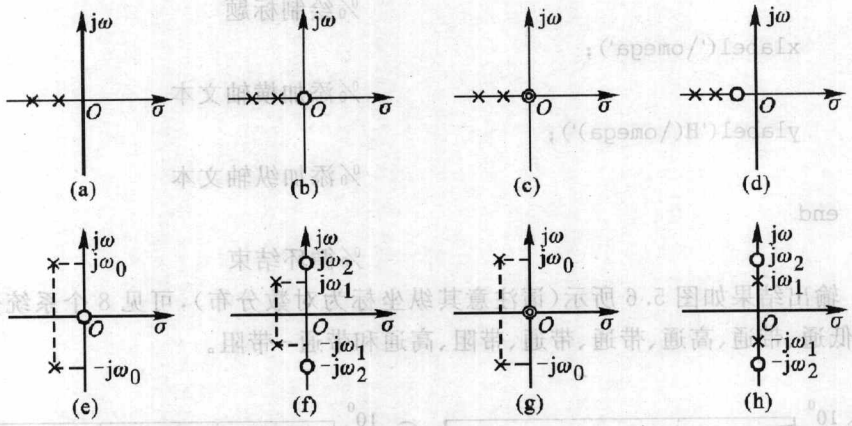


图 5.5　例 5.8 系统零、极点分布图

```
'(d)','(e)','(f)','(g)','(h)'},'zeros',{[],...
                                    %数组中每个元素都是结构
    [0],[0;0],[-0.5],[0],[1.2j;-1.2j],[0;0]},...
                                    %每个结构有三个域分别是
    [1.2j;-1.2j],'poles',{[-2;-1],[-2;-1],...
                                    %'title','zeros'和'poles'
    [-2;-1],[-2;-1],[-1+j;-1-j],[-1+j;-1-j],...
    [-1+j;-1-j],[j;-j]});
omega=[0:0.01:6];                   %生成频率抽样点
figure;                             %生成新图框
for id=1:8                          %依次绘制 8 个系统的频响
    [b,a]=zp2tf(data(id).zeros,data(id).poles,1);
                                    %由零、极点得到传递函数系数
    H=freqs(b,a,omega);             %计算指定频率点的响应 H(ω)
    subplot(4,2,id);                %生成子图
    plot(omega,abs(H));             %绘制频响
    set(gca,'YScale','log','FontSize',16);
                                    %设 Y 轴为对数坐标,16 号字体
    title(data(id).title);
```

```
                                        %绘制标题
    xlabel('\omega');
                                        %添加横轴文本
    ylabel('H(\omega)');
                                        %添加纵轴文本
end
                                        %循环结束
```

输出结果如图 5.6 所示(请注意其纵坐标为对数分布),可见 8 个系统分别是:低通、带通、高通、带通、带通、带阻、高通和带通-带阻。

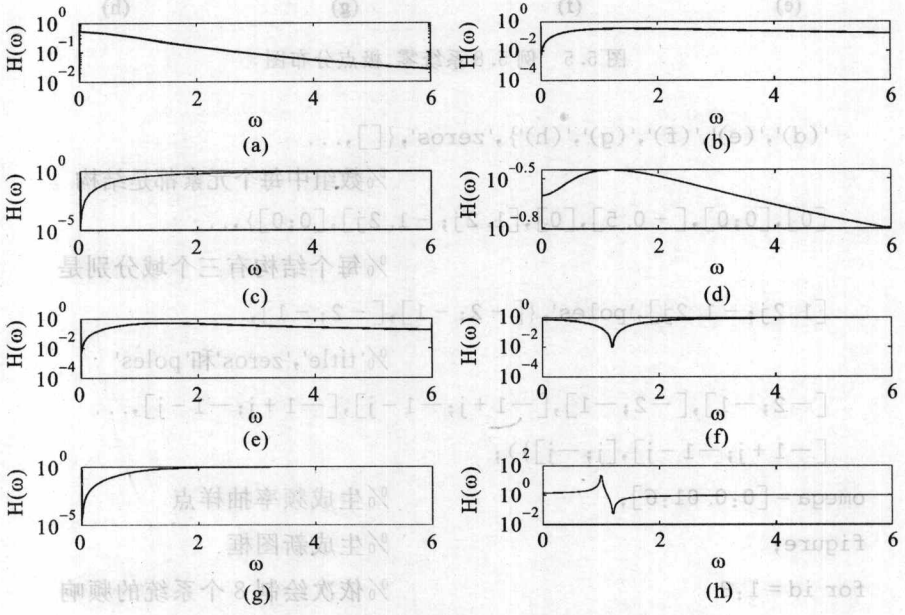


图 5.6 例 5.8 运行结果

MATLAB 知识点(10)——结构

类似于 C 语言中的结构,MATLAB 中的结构类型也是一种可以把不同类型的数据组合在一起的数据结构;和 C++的类一样,MATLAB 中用操作符“.”引用结构的成员变量(称作“域”)。用 struct 函数定义结构,比如

```
stu = struct('name',{'Tom'},'age',{6})
```

将建立一个包括两个域“name”和“age”的结构变量 stu,并且将这两个域分别初

始化为“Tom”和 6，接下来就可以用“stu. name”和“stu. age”分别引用这两个域。结构数组用类似的方法建立，比如

```
stus = struct('name',{'Tom','Jerry'},'age',{6,5})
```

建立了一个包括两个元素的结构数组“stus”，引用第二个元素的域的方式为“stus(2). name”。

如果对结构初始化时建立的域不足，还可以随时补充，比如就上例中，如果再输入

```
stus(1).gender = 'male'
```

则此结构除了“name”和“age”外，又增加了一个域“gender”，并且“name”为“tom”的元素的“gender”域赋值为“male”，而另一个“name”为“Jerry”的元素的“gender”域则为空(“[]”)。

MATLAB 还提供了诸如 isstruct 和 isfield 等函数实现判断结构和域的属性等各种基本操作，从而保证利用结构可以实现复杂的数据管理功能。

第五节　二阶谐振系统的 s 平面分析

根据主教材公式(4-119)和公式(4-122)，将二阶谐振系统写成标准形式

$$Z(s)=\frac{s}{s^2+2\alpha s+\omega_0^2}$$

下面例题由 α 和 ω_0 的相对关系考察其时频特性。

例 5.9(主教材图 4-37 修改)　一个并联谐振电路，极点分布随 α 增大由虚轴上沿单位圆旋转到负实轴，绘制电路系统的零、极点分布图，冲激响应，幅频响应和相频响应。

解　将设计一个动画程序，希望程序运行后动态演示一对共轭极点从虚轴旋转到负实轴的过程中的零、极点分布图，冲激响应和频率响应的变化。

§§§MATLAB 文件 ex_5_9. m

```
omega0 = 1;                                  %定义 ω0=1
b = [1,0];                                   %传递函数分子多项式
                                             %系数不变
for theta = pi/2:pi/20:pi                    %循环，θ从 π/2 逐渐增
                                             %大到 π
  alpha = omega0 * - cos(theta);             %计算 α
  a = [1,2 * alpha,omega0^2];                %定义传递函数分母多
```

```
                                        %项式系数
    figure(1);                          %生成(激活)图框 1
    subplot(2,1,1);                     %生成左上角的子图框
    pzmap(b,a);                         %绘制零极点图
    set(gca,'XLim',[-1,1],'YLim',[-1,1]);  %设置图框视野
    subplot(2,1,2);                     %绘制左下角的子图框
    impulse(b,a);                       %绘制冲激响应
    figure(2);                          %生成(激活)图框 2
    freqs(b,a);                         %绘制幅度和相位的频
                                        %率响应
    pause(0.2);                         %暂停,迫使绘图事件
                                        %发生
end                                     %循环结束
```

上例中的 pause(s)语句暂停程序 s 秒,以便绘图函数执行结束并在屏幕上显示出绘图效果来。

第六节 小结

拉氏变换有两个重要用途。首先用于解微分方程,连续时间域复杂的微积分运算在拉氏变换域表现为简单的多项式乘除法。因而由拉氏变换式逆变换回时域表达式就成了解微分方程的关键问题。MATLAB 提供了实现部分分式展开的 residue 函数,可计算出自由项以及各个分式的系数和极点,然后即可手工写出时域表达式。拉氏变换的另一个重要用途是根据零、极点的分布考察系统的时域响应和频域响应。函数 pzmap 和 freqs 分别用于绘制零、极点分布图和频率响应曲线,而时域响应仍然用上一章介绍的 impulse 函数绘制。

练习题

1. (主教材习题 4-4 修改)对下面两个系统函数

$$H_1(t)=\frac{4s+5}{s^2+5s+6}$$

$$H_2(t)=\frac{1}{s^2+1}+1$$

分别用 residue 计算冲激响应理论值,再和用 lsim 仿真得到的冲激响应比较是否相同。

2. 编写函数 s=isstable(sys)，判断用传递函数描述的系统模型 sys 的稳定性。返回值 s 为 1 表示系统稳定，为 0 表示不稳定。分别设计一个稳定系统和一个不稳定系统验证该函数的正确性。

3.（主教材习题 4－27 修改）已知激励信号 $e(t)=\mathrm{e}^{-t}$，零状态响应 $r(t)=\frac{1}{2}\mathrm{e}^{-t}-\mathrm{e}^{-2t}+2\mathrm{e}^{-3t}$，绘制此系统的冲激响应 $h(t)$、频率响应 $H(\mathrm{j}\omega)$ 和零、极点分布图，利用 isstable 函数判断其稳定性。

第六章　音乐合成

本章中将基于傅里叶级数和傅里叶变换等基础知识，应用第一篇讲授的 MATLAB 编程技术，在电子音乐合成领域做一些练习。通过本章的练习，可以增进对傅里叶级数的理解，熟练运用 MATLAB 基本指令。本章包括两部分，第一部分介绍乐理和电子音乐的基本知识，第二部分给出详细的练习内容和编程步骤。相信读者对此会产生强烈兴趣。

第一节　背景知识

6.1.1　乐音特征

人类听觉可以感受到的声音大体上可划分为噪声、语音、乐音等几种类型。关于噪声和语音的特征及其性能分析将在本科高年级或研究生的课程中专门研究。本练习初步介绍乐音的基本概念，并利用 MATLAB 编程实现音乐合成系统。

音乐是乐音随时间流动而形成的艺术。用通信与电子技术的术语解释就是周期信号频率（某种指定规律的频谱结构）随时间节奏变化的一种表述。乐谱上的每个音符表达了此时此刻规定出现的信号频率和持续时间。

乐音的基本特征可以用基波频率、谐波频谱和包络波形三个方面来描述，下面将分别说明。

很明显，认识乐音的频谱规律之后，就可以借助电子系统从软件或硬件两种角度模仿各种乐器产生的声音，实现所谓电子音乐系统。

电子音乐系统是一门新兴的交叉技术科学，涉及计算机、集成电路、电子线路、信号处理、声学等多种领域，研究与应用前景广阔而深远。本练习只是非常简单的入门介绍。

6.1.2　乐音基波构成规律

用 C、D、E、F、G、A、B 大写英文字母表示每个音的“音名”（或称为“音调”），

当指定某一音名时，它对应固定的基波信号频率。

图 6.1 所示为钢琴键盘结构，并注明了每个琴键对应的音名和基波频率值。这些频率值按“十二平均律”计算导出，下面解释计算规则。

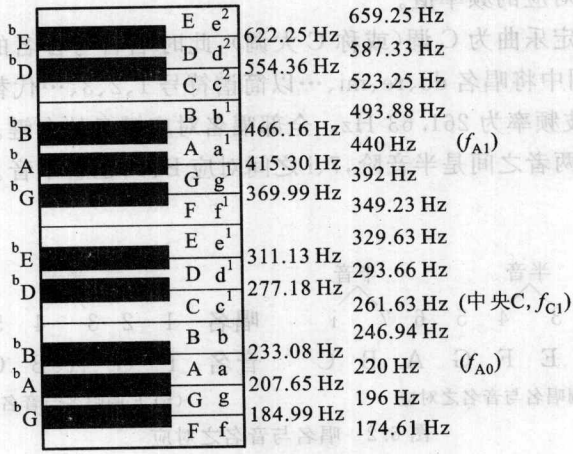


图 6.1　钢琴键盘和相应频率

从图 6.1 可以看到，靠下边的 A 键称为小字组 a，它的频率值 $f_{A0}=220$ Hz。而靠上面的另一个 A 键是小字一组 a^1，它的频率值是 $f_{A1}=440$ Hz。两者为二倍频率关系，即 f_{A1} 相当于 f_{A0} 的二次谐波。也称为 8 度音程或倍频程 Octave（即画频响特性波特图时所用的术语“倍频程”）。

从小字组 a 到小字一组 a^1 之间共有 12 个键，其中 7 个白色键，5 个黑色键，其频率值计算规律为相邻音倍乘系数 $K=2^{\frac{1}{12}}=1.059\,463\,09$。

由此可求出图中各琴键对应之频率值。例如从 f_{A0} 导出小字一组 c^1（也称为中央 C）的频率为

$$f_{C1}=220\times 2^{\frac{3}{12}}\text{Hz}=261.63\text{ Hz} \tag{6.1}$$

或利用 f_{A1} 也可导出同样结果。

人耳听觉辨识振动频率的能力大约在±0.5%，上述规律刚好符合这一要求。

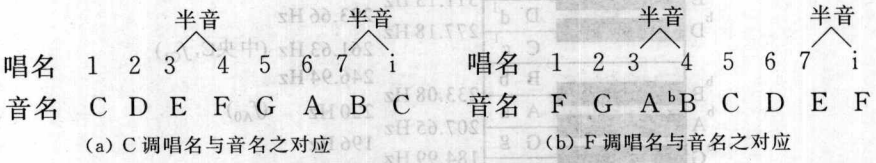
从图 6.1 可以看出 7 个白键之间插入了 5 个黑键。在 EF 之间和 BC 之间没有黑键，也即这两组相邻的白键之间基波频率倍乘系数为 $2^{\frac{1}{12}}$，也称为相隔半音，而在其他白键之间都有黑键相隔，因而它们的频率倍乘系数为 $2^{\frac{2}{12}}$，也称为相隔全音（如 CD、DE、FG、…之间）。若以白键英文字母为基准，则升高半音以“♯”符号表示，降低半音则以“b”符号表示。于是，可以写出 12 个音名从低到高的依次字母表示为

C,$^{\flat}$D,D,$^{\flat}$E,E,F,$^{\flat}$G,G,$^{\flat}$A,A,$^{\flat}$B,B,C

当然，若改用“#”号表示黑键，则$^{\flat}$D改为$^{\#}$C，$^{\flat}$E改为$^{\#}$D，…。

下面给出“唱名”的概念。所谓唱名是指平日读乐谱唱出的do、re、mi、…。每个唱名并未固定基波频率。当指定乐曲的音调时才知道此时唱名对应的音名，也即确定了对应的频率值。

例如，若指定乐曲为C调(或称C大调)，此时唱名与音名的对应关系如图6.2(a)所示。图中将唱名do、re、mi、…以简谱符号1、2、3、…代替。可见，它的1对应C也即基波频率为261.63 Hz。全部唱名对应键盘的白键。还可看出，3、4分别对应E、F，两者之间是半音阶，7、i之间对应B、C，也属半音。其他唱名之间都为全音。



(a) C调唱名与音名之对应　　(b) F调唱名与音名之对应

图6.2 唱名与音名之对应

如果改为F调(F大调)，唱名与音名的对应关系如图6.2(b)所示。它的1与F对应，频率值是349.23 Hz。为了保持3、4之间为半音之规律，4对应黑键$^{\flat}$B。

为了保持3、4以及7、i之间为半音的规律，只有C调的全部唱名都与白键对应。而其他各调都需要引用黑键。如上例F调的4对应$^{\flat}$B，其他各调对应黑键的规律可作为练习请读者自行导出。

下面给出一个乐曲实例，练习写出每个唱名对应的基波频率值，如图6.3所示。这是《东方红》的开头两句曲谱，用简谱写出。曲调定为F，即1=F，于是可查出第一个音5对应C，频率值为523.25 Hz，其他音的频率可依次写出。稍后，将以此为基础进行音乐合成练习。

$$1=\mathrm{F}\ \frac{2}{4}\quad |\ 5\quad \underset{\frown}{56}\ |\ 2\ -\ |\ 1\ \underset{\frown}{\underset{\cdot}{16}}\ |\ 2\ -\ |$$

图6.3 乐曲《东方红》第一小节曲谱

6.1.3 乐音谐波的作用——音色

当指定音名(音调)之后仅指定了乐音信号的基波频率，谐波情况并未说明。对于各种乐器如钢琴或单簧管都可发出$f_{A1}=440$ Hz的乐音，而人的听觉会明显感觉两者不同，这是由于谐波成分有所区别，频谱结构各异。例如单簧管的三次、五次谐波成分很强，其他各种乐器都有自己的谐波分布规律。同种乐器不同音阶的谐波构成还可能略有区别。由于演奏技巧、方法的差异也可产生不同结

构的谐波。

在制作电子乐器(如电子琴)时,应尽力模仿实际乐器的谐波结构。但是由于人为因素的随机变化,往往感觉电子琴产生的乐音与利用传统乐器产生的乐音很难完全一致。

在音乐领域中称谐波为"泛音"。由谐波产生的作用称为音色变化。

在稍后的编程练习中将要看到,如果只考虑乐音的基波成分,每个音名对应不同频率的正弦(余弦)波;当引入谐波分量之后,波形不再是简单的正弦函数,例如,可能接近矩形、锯齿波等。

6.1.4 乐音波形包络

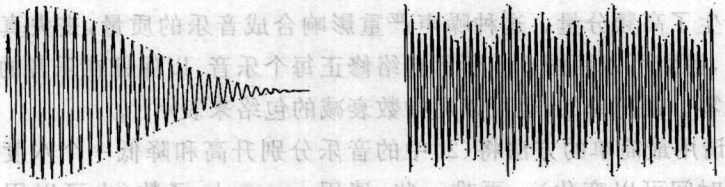
这是描述乐音特性的另一个重要因素。除了前述基频和谐波表征了乐音特性之外,对于不同类型的乐器它们的包络形状也不相同,在电子乐器制作中习惯上称此包络为"音型"或"音形"。实际的波形好像通信系统中经过调制的信号。

各种乐器的包络(音型)大体上可划分为以下几种类型:

- 连续型(风琴、手风琴、弦乐)
- 弹奏型(钢琴、吉他)
- 拨奏型(琵琶、月琴、曼多林)
- 击奏型(木琴、木鱼)
- 吹奏型(管乐)
- 颤音型(调频信号,如小提琴揉音)

图 6.4 所示为钢琴与管乐的波形,可以看出两者包络的区别。其他类型省略。

在乐音合成实验中,为简化编程描述,通常把复杂的包络函数用少量直线近似,于是,乐音波形的包络呈折线。



(a) 钢琴　　(b) 管乐

图 6.4　钢琴和管乐的信号波形

6.1.5 音调持续时间

在最简单的例子中,每个音调都可以用连续的一段正弦信号并带有一小段

静音(停顿)来表示。停顿保证我们可以区分开连续的相同音调。每个音调的持续时间取决于它是全音符、二分之一音符、四分之一音符还是八分之一音符等等。很明显,四分之一音符的持续时间是八分之一音符的两倍。而每个音符之后的停顿时间应该是相同的,不随音符的长度而变化。在乐谱中,更长一些的停顿要用休止符来表示。图 6.3 中左侧的$\frac{2}{4}$表示每小节有二拍,一个四分之一音符的持续时间为一拍。对这段乐曲来说,一拍大约是 0.5 s。

6.1.6 音符的叠接

某些艺术家演奏乐器时相邻的音符会有些重叠,就是当一个音调消失的时候,另一个被演奏出来。数学的表示就是两个信号有些重叠。这样听起来会更连续,较少断音。注意:这里说的叠接和前面要求的停顿并不矛盾,叠接时两个信号的幅度差别必须足够大以保证被区分开。

第二节 音乐合成综合实验

6.2.1 简单的合成音乐

(1) 请根据《东方红》第一小节的简谱和"十二平均律"计算出该小节中各个乐音的频率,在 MATLAB 中生成幅度为 1,抽样频率为 8 kHz 的正弦信号表示这些乐音。请用 sound 函数播放每个乐音,听一听音调是否正确。最后用这一系列乐音信号拼出《东方红》第一小节,注意控制每个乐音持续的时间要符合节拍,用 sound 播放你合成的音乐,听起来感觉如何?

(2) 你一定注意到(1)的乐曲中相邻乐音之间有"啪"的杂声,这是由于相位不连续产生了高频分量。这种噪声严重影响合成音乐的质量,丧失真实感。为了消除它,我们可以用图 6.5 所示包络修正每个乐音,以保证在乐音的邻接处信号幅度为零。最简单的,也可以用指数衰减的包络来表示[①]。

(3) 请用最简单的方法将(2)中的音乐分别升高和降低一个八度(提示:音乐播放的时间可以变化)。再难一些,请用 resample 函数(也可以用 interp 和 decimate 函数)将上述音乐升高半个音阶(提示:视计算复杂度,不必特别精确)。

(4) 试着在(2)的音乐中增加一些谐波分量,听一听音乐是否更有"厚度"了?注意谐波分量的能量要小,否则掩盖住基音反而听不清音调了。(我选择基波幅度为 1,二次谐波幅度 0.2,三次谐波幅度 0.3,听起来有点像风琴。)

① 根据人耳听觉特性,当主观感受声强为线性变化时,声音信号的功率实际上呈指数变化。

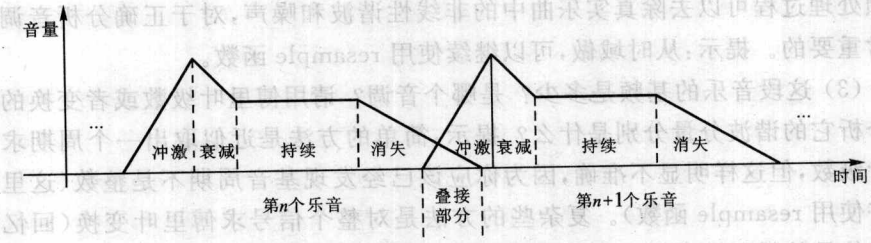


图 6.5　音量变化

(5) 自选其他音乐合成，例如《贝多芬第五交响乐》的开头两小节。

6.2.2　用傅里叶级数分析音乐

现在要开始处理真实的音乐信号了！请用 load 命令载入中国高校电工电子课程网上的数据文件 guitar. mat，工作区会出现两个新的变量 realwave 和 wave2proc(可以用 who 查看变量名)，如图 6.6 和图 6.7 所示。其中前者是从一段吉他乐曲中截取下来的真实信号，后者是用信号处理方法得到的这段信号的理论值，它们的抽样率都是 8 kHz。

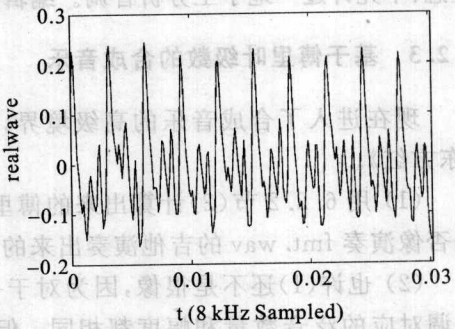


图 6.6　真实吉他音

(1) 先用 wavread 函数载入中国高校电工电子课程网上的 fmt. wav 文件，播放出来听听效果如何？是否比刚才的合成音乐真实多了？

(2) 你知道待处理的 wave2proc 是如何从真实值 realwave 中得到的吗？这

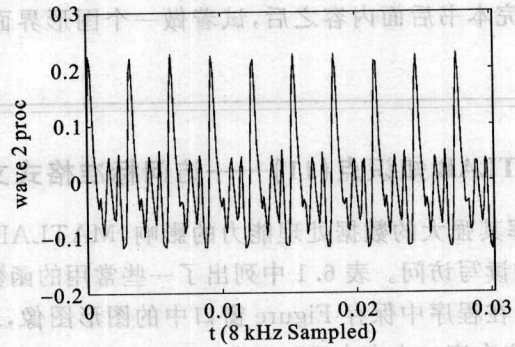


图 6.7　待处理的吉他音

个预处理过程可以去除真实乐曲中的非线性谐波和噪声，对于正确分析音调是非常重要的。提示：从时域做，可以继续使用 resample 函数。

(3) 这段音乐的基频是多少？是哪个音调？请用傅里叶级数或者变换的方法分析它的谐波分量分别是什么？提示：简单的方法是近似取出一个周期求傅里叶级数，但这样明显不准确，因为你应该已经发现基音周期不是整数(这里不允许使用 resample 函数)。复杂些的方法是对整个信号求傅里叶变换(回忆周期性信号的傅里叶变换)，但你可能发现无论你如何提高频域的分辨率，也得不到精确的包络(应该近似于冲激函数而不是 sinc 函数)。可选的方法是增加时域的数据量，即再把时域信号重复若干次，看看这样是否效果好多了？请解释之。

(4) 再次载入 fmt. wav，现在要求编写一段程序，自动分析出这段乐曲的音调和节拍。如果觉得太难，允许手工标定出每个音调的起止时间，还可以把每个音调的数据都单独保存成一个文件，然后让 MATLAB 对这些文件进行批处理。注意：不允许逐一地手工分析音调。编辑音乐文件，推荐使用 CoolEdit 编辑软件。

6.2.3 基于傅里叶级数的合成音乐

现在进入了合成音乐的高级境界，要用演奏 fmt. wav 的吉他合成出一段《东方红》。

(1) 用 6.2.2 节(2)计算出来的傅里叶级数再次完成 6.2.1 节(4)，听一听是否像演奏 fmt. wav 的吉他演奏出来的？

(2) 也许(1)还不是很像，因为对于一把泛音丰富的吉他而言，不可能每个音调对应的泛音数量和幅度都相同。但是通过完成 6.2.2 节(4)，已经提取出 fmt. wav 中的很多音调，或者说，掌握了每个音调对应的傅里叶级数，大致了解了这把吉他的特征。现在就来演奏一曲《东方红》吧。提示：如果还是音调信息不够，那就利用相邻音调的信息近似好了，毕竟可以假设吉他的频响是连续变化的。

(3) 现在只要掌握了某乐器足够多的演奏资料，就可以合成出该乐器演奏的任何音乐，在学完本书后面内容之后，试着做一个图形界面把上述能力封装起来。

MATLAB 知识点(11)——访问标准格式文件

为了充分发挥其强大的数据处理能力的影响，MATLAB 支持对工业界多种标准格式文件的读写访问。表 6.1 中列出了一些常用的函数。

为了支持用户在程序中保存 Figure 窗口中的图形图像，MATLAB 提供了 saveas 函数，和用户在窗口中点击 File→Save As 一样，它支持的包括 PDF 在内的图像文件格式有 16 种之多！

表 6.1　访问标准格式文件的常用函数

函　数　名	说　　明
dlmread dlmwrite textscan，csvread csvwrite 等	访问各种 ASCII 文本格式文件
xlsread，xlswrite	访问微软 EXCEL 格式文件
wavread，wavwrite	访问微软 WAV 格式音频文件
auread，auwrite	访问 NeXT/SUN 格式音频文件
imread，imwrite	访问 JPEG，TIFF，GIF，BMP 等多种格式图像文件
aviread，avifile 等	访问 AVI 视频文件

表 6.1 近两年常用文件的常用函数

函 数 名	说 明
dlmread dlmwrite textscan, csvread csvwrite 等	访问各种 ASCII 文本格式文件
xlsread, xlswrite	访问微软 EXCEL 格式文件
wavread, wavwrite	访问微软 WAV 格式声频文件
auread, auwrite	访问 NeXT/SUN 格式声频文件
imread, imwrite	访问 JPEG, TIFF, GIF, BMP 等多种格式图像文件
aviread, avifile 等	访问 AVI 视频文件

第三篇

离散时间信号与系统

第三篇

离散时间信号与系统

第七章　离散时间系统的时域分析

本章首先讲授求解常系数线性差分方程的方法，然后介绍如何绘制离散时间系统单位样值响应。接下来以一道例题演示三种计算卷积和的方法，掌握这些方法有助于理解离散时间系统。最后介绍卷积和的逆运算——解卷积。

本章包括两个 MATLAB 知识点。① 处理列矢量；② 多项式相乘与卷积和的关系。

学习重点：filter 函数的用途和多种使用方法。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
filter	离散时间系统仿真，数字滤波	impz	计算单位样值响应
deconv	解卷积	mod	取模

第一节　常系数线性差分方程的求解

MATLAB 提供数值解法（即迭代法）求差分方程的完全解，即给定传递函数、激励序列和边界条件后，用 filter 函数得到输出序列。对于用差分方程

$$y(n)+a_1y(n-1)+\cdots+a_Ny(n-N)=b_0x(n)+b_1x(n-1)+\cdots+b_Mx(n-M)$$

所描述的离散时间系数，[y,wf]=filter(b,a,x,wi)用于计算激励 $x(n)$ 作用下的响应 $y(n)$，其中 a,b 对应 $\boldsymbol{a}=[1,a_1,a_2,\cdots,a_N]^{\mathrm{T}}$，$\boldsymbol{b}=[b_0,b_1,\cdots,b_M]^{\mathrm{T}}$ 分别表示差分方程左侧和右侧的系数。wi 和 wf 分别表示系统的初始状态和终止状态，上式离散系统可以用图 7.1 结构实现，图中 z^{-1} 为延时单元，$\sum$ 实现相加运算，假设 $M=N$，其中 $w_i(n)$，$i=1,2,\cdots,M$，此函数为中间变量，有

$$w_M(n)=b_Mx(n)-a_My(n)$$

$$w_{M-1}(n)=w_M(n-1)+b_{M-1}x(n)-a_{M-1}y(n)$$

……

$$w_1(n)=w_2(n-1)+b_1x(n)-a_1y(n)$$

$$y(n)=w_1(n-1)+b_0x(n)$$

综合以上各公式,得到

$$y(n)=\sum_{k=0}^{M}b_kx(n-k)-\sum_{j=1}^{N}a_ky(n-k)$$

即待求差分方程。图 7.1 中各个延时单元的状态则对应于 wi 和 wf。即 $\boldsymbol{w}_i=[w_1(-1),w_2(-1),\cdots,w_M(-1)]^{\mathrm{T}}$,$\boldsymbol{w}_f=[w_1(L-1),w_2(L-1),\cdots,w_M(L-1)]^{\mathrm{T}}$,注意状态变量的个数等于 M 和 N 中的较大者,另外假设数据 $x(n)$ 长度为 L。

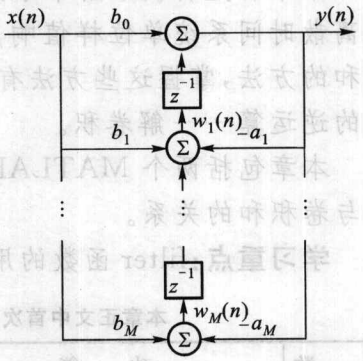


图 7.1 滤波器实现结构和状态变量

此结构之含义在主教材第七、八章有初步介绍,第十章 10.6 节和第十一章习题 11-32 有更详细的研究。在后续课程"数字信号处理"中会得到更多应用。

例 7.1(主教材例 7-9) 求下示差分方程的完全解

$$y(n)+2y(n-1)=x(n)-x(n-1)$$

其中激励函数 $x(n)=n^2$,且已知 $y(-1)=-1$。

解 §§§MATLAB 文件 ex_7_1.m

```
a = [1,2];                    %定义差分方程左侧系数矢量 a
b = [1, -1];                  %定义差分方程右侧系数矢量 b
n = [0:20];                   %生成仿真序列时间 n
x = n.^2;                     %生成激励信号 x(n)
wi = b(2) * 0 - a(2) * ( - 1);  %定义初始状态 wi=b(2) * x(-1)-
                              %a(2) * y(-1)①
[y wf] = filter(b,a,x,wi);    %滤波得到输出序列 y(n)
ex_7_1_plot();                %绘制输出序列
```

运行结果如图 7.2 所示。

例 7.2(主教材例 7-10) 已知系统的差分方程表达式为

① 注意 $x(-1)=0$。

$$y(n)-0.9y(n-1)=0.05u(n)$$

① 若边界条件 $y(-1)=0$，求系统的完全响应；② 若边界条件 $y(-1)=1$，求系统的完全响应。

解　下面分别绘出两种情况下的零输入响应、零状态响应和完全响应。因为要对两个边界条件做相同处理，所以可将其以两个列矢量的形式拼成矩阵，然后调用一次 filter 函数即可。

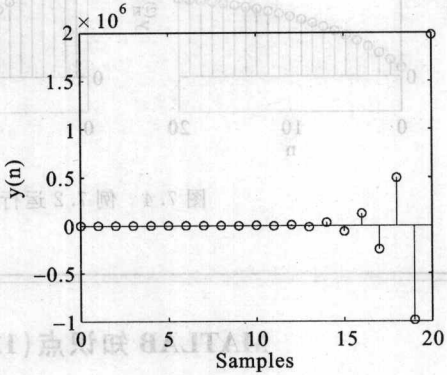


图 7.2　例 7.1 运行结果

§§§MATLAB 文件 ex_7_2.m

```
a=[1,-.9];
b=.05;
n=[-5:20]';
u=(n>=0);                                    %生成激励序列 u(n)
uz=zeros(1ength(u),1);                       %生成零输入信号
wil=-a(2)*0;                                 %定义两个初始状态
wi2=-a(2)*1;
[yzs wf]=filter(b,a,[u,u]);                  %滤波得到零状态响应
[yzi wf]=filter(b,a,[uz,uz],[wil,wi2]);      %滤波得到零输入响应
[y wf]=filter(b,a,[u,u],[wil,wi2]);          %滤波得到完全响应
ex_7_2_plot();                               %绘制输出图形
```

运行结果如图 7.3 和图 7.4 所示。

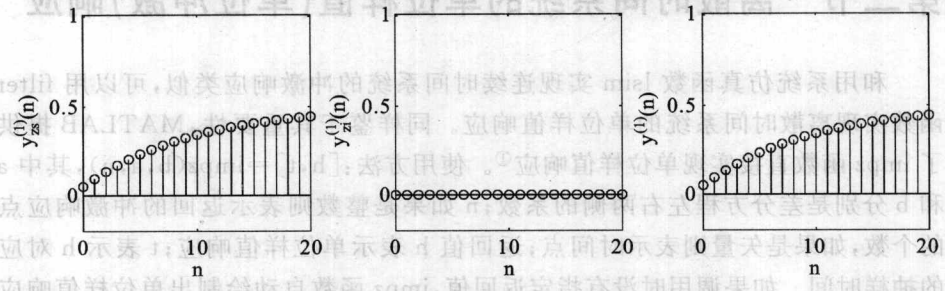


图 7.3　例 7.2 运行结果(边界条件一)

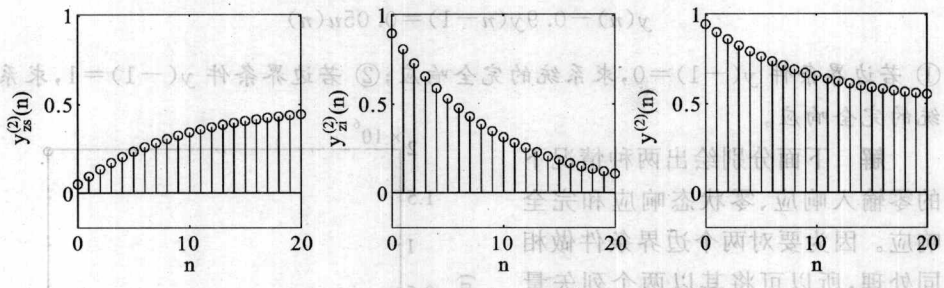


图 7.4 例 7.2 运行结果(边界条件二)

MATLAB 知识点(12)——处理列矢量

对大部分处理一维信号(矢量)的函数,例如 filter,如果以二维信号(矩阵)做输入,MATLAB 会把它当成多个一维信号同时进行处理,并且输出也跟随输入的格式和顺序。和数字信号处理理论中常用的表达方法一样,MATLAB 默认信号以列矢量形式表示,虽然它并不强制要求用户这么做。例如 filter 函数的激励信号可以是行矢量也可以是列矢量,而且响应会服从激励的格式。但当输入一个矩阵并且没有约定信号所在维数的时候,会看到它被当成多个列矢量处理了。

不仅信号处理工具箱采用默认列矢量的约定,事实上 MATLAB 的主包也是如此。比如执行逻辑运算的 all 函数:如果输入矩阵 X 并且没有声明计算哪一维的话,all(X)将返回一个行矢量,其中每个元素表明了 X 中的该列是否为全 1。

第二节 离散时间系统的单位样值(单位冲激)响应

和用系统仿真函数 lsim 实现连续时间系统的冲激响应类似,可以用 filter 函数实现离散时间系统的单位样值响应。同样鉴于其重要性,MATLAB 提供了 impz 函数直接实现单位样值响应[①]。使用方法:[h,t]=impz(b,a,n),其中 a 和 b 分别是差分方程左右两侧的系数;n 如果是整数则表示返回的冲激响应点的个数,如果是矢量则表示时间点;返回值 h 表示单位样值响应;t 表示 h 对应的抽样时间。如果调用时没有指定返回值,impz 函数自动绘制出单位样值响应的波形。

① 绘制离散时间系统阶跃响应的函数是 stepz。

例 7.3(主教材例 7-14 修改)　已知系统的差分方程模型

$$y(n)-0.5y(n-1)+0.6y(n-2)=x(n)-0.3x(n-2)$$

求系统的单位样值响应。

解　分别用 filter 和 impz 计算单位样值响应。

```
§§§MATLAB 文件 ex_7_3.m
a=[1,-5,6];
b=[10,-3];
n=[0:10]';
[hi,t]=impz(b,a,n);          %用 impz 函数计算冲激响应 hi(n)
x=(n==0);                    %以单位样值序列为激励信号
hf=filter(b,a,x);            %经 filter 函数滤波得到 hf(n)
ex_7_3_plot();               %绘制输出图形
```

运行结果如图 7.5 所示，可见两种方法得到的结果完全相同。

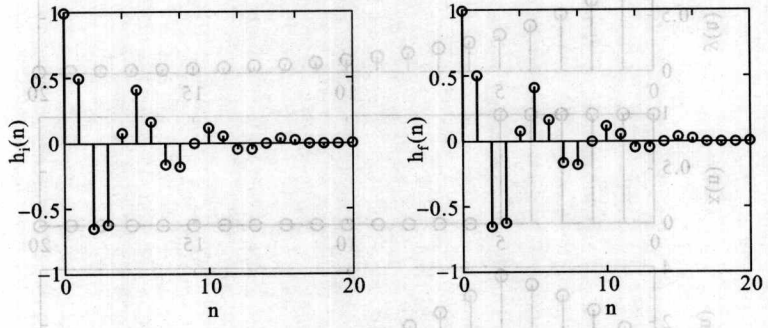


图 7.5　例 7.3 运行结果

第三节　卷积(卷积和)

在第三章第五节讲过用 conv 函数和数值方法实现卷积积分，事实上，conv 函数是对两个多项式做相乘，因而可以用来计算离散时间卷积。另外，根据筛选特性，序列通过线性系统就是序列和系统单位样值响应进行卷积，因而卷积运算也可以用 filter 函数实现，即将两个待卷积序列分别理解为系统单位样值响应和激励序列。

例 7.4(主教材例 7-15 修改)　某系统的单位样值响应是 $h(n)=a^nu(n)$，

其中 $a=0.8$。若激励信号为 $x(n)=u(n)-u(n-6)$，试求响应 $y(n)$。

解 分别用 conv 和 filter 求卷积结果，注意该系统的冲激响应是无穷长的。若用 conv 函数，需要将 $h(n)$ 截断，然后直接卷积即可；若用 filter 函数，必须将系统写成差分方程的表示形式(为了提取 a、b 系数)，这样又有两种方法，简单的方法是先将系统响应和激励交换，即认为系统响应是 $h_1(n)=x(n)=u(n)-u(n-N)$，从而直接看出 $\boldsymbol{b}_1=[1,1,\cdots,1]^{\mathrm{T}}$($N$ 个 1)，$\boldsymbol{a}_1=1$，然后再用 filter 函数对 $x_1(n)=h(n)=a^n u(n)$ 执行滤波操作。复杂些的方法必须在学过 z 变换后才能使用，即由冲激响应 $h(n)$ 可知系统 z 变换为 $H(z)=\dfrac{1}{1-az^{-1}}$，从而知道系统的差分方程表示为 $y(n)-ay(n-1)=x(n)$，即 $\boldsymbol{b}=1$，$\boldsymbol{a}=[1,-a]^{\mathrm{T}}$。下面用 MATLAB 分别实现这三种方法。

§§§MATLAB 文件 ex_7_4.m

```
a = 0.8;                          %定义系数 a=0.8
```

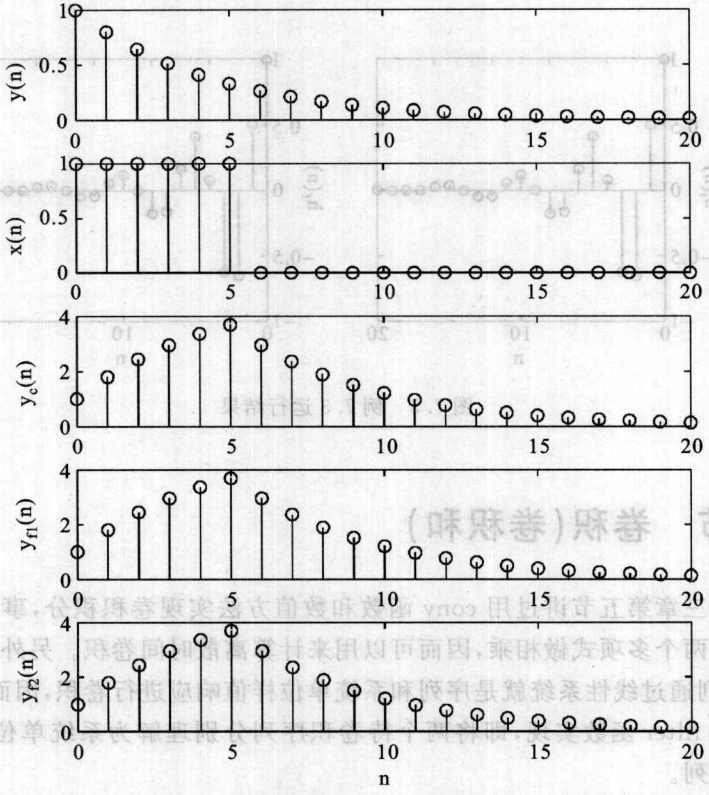


图 7.6 例 7.4 运行结果

```
N = 6;                          %定义激励长度 N=6
n = [0:20]';                    %定义时间
h = (a.^n). * (n> = 0);         %生成单位样值响应 h(n)
x = (n> = 0 & n<N);             %生成激励信号 x(n)
yc = conv(h,x);                 %用 conv 计算卷积,注意 yc 长度变化
yf1 = filter(x,1,h);            %用 h(n)作为激励
yf2 = filter(1,[1, - a],x);     %用 x(n)作为激励
ex_7_4_plot();                  %生成新图框并在子图内分别绘制各个序列
```

运行结果如图 7.6 所示，可见三种方法得到的结果完全相同。

MATLAB 知识点(13)——多项式相乘和卷积的关系

假设有两个多项式 $p_a(x)$和 $p_b(x)$，其乘积

$$p_a(x)p_b(x)=\sum_{i=0}^{n}a_ix^{n-i}\sum_{j=0}^{m}b_jx^{m-j}=\sum_{i=0}^{n}\sum_{j=0}^{m}a_ib_jx^{n+m-i-j}$$

定义 $k=i+j$，替换掉下标 j，则有

$$p_a(x)p_b(x)=\sum_{i=0}^{n}\sum_{k=i}^{m+i}a_ib_{k-i}x^{n+m-k}=\sum_{k=0}^{n+m}\sum_{i=\max(0,k-m)}^{\min(n,k)}a_ib_{k-i}x^{n+m-k}$$

对比用单一系数符号表达的形式 $p_a(x)p_b(x)=\sum_{k=0}^{n+m}c_kx^{n+m-k}$，可见

$$c_k=\sum_{i=\max(0,k-m)}^{\min(n,k)}a_ib_{k-i}$$

即序列卷积和的定义，所以多项式乘法中系数的变换就对应于卷积和计算。

第四节　解卷积(反卷积)

MATLAB 提供[q,r]=deconv(b,a)函数实现解卷积，其中 b=conv(a,q)+r，即 a 和 q 卷积后再加上余量 r 得到 b，而解卷积就是根据 b 和 a 解出 q 和 r。下面通过原著一道习题演示解卷积的用法和功能。

例 7.5(主教材习题 7-35)　某地址勘探测试设备给出的发射信号 $x(n)=\delta(n)+\frac{1}{2}\delta(n-1)$，接收回波信号 $y(n)=\left(\frac{1}{2}\right)^n u(n)$，若地层反射特性的系统函

数以 $h(n)$ 表示，且满足 $y(n)=h(n)*x(n)$。① 求 $h(n)$；② 以延时、相加、倍乘运算为基本单元，试画出系统方框图。

解 利用主教材介绍的解卷积方法，可求出

$$h(n)=\begin{cases}\left(\frac{1}{2}\right)^n & n\text{ 为奇数}\\ 0 & n\text{ 为偶数}\end{cases}$$

下面用 deconv 函数进行验证

§§§MATLAB 文件 ex_7_5.m

```
n = [0:20];                          %定义抽样时间
x = [1;0.5];                         %定义发射信号
y = 0.5.^n;                          %定义回波信号
[h,r] = deconv(y,x);                 %解卷积求单位样值响应
h0 = 0.5.^n. * (mod(n,2) == 0);      %定义理论结果做比较
ex_7_5_plot();
```

程序运行结果如图 7.7 所示，可见 deconv 的输出和理论值基本相同。系统方框图如图 7.8 所示。

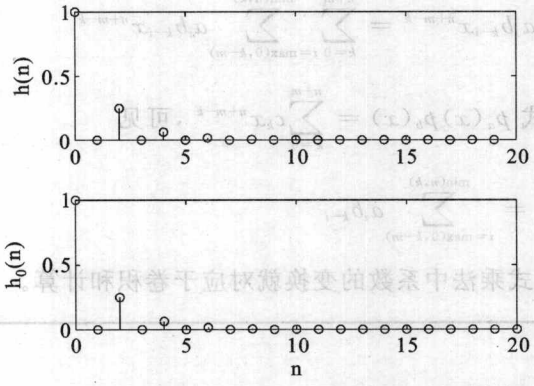


图 7.7 例 7.5 运行结果

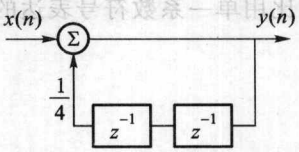


图 7.8 例 7.5 系统方框图

第五节 小结

本章介绍了从时域出发对离散时间系统进行描述、分析和求解的方法。对应于连续时间系统仿真的 lsim 函数，filter 函数用于求解差分方程和仿真离散时间系统。差分方程的边界条件在该函数中表现为初始状态，也即 filter 函数

用状态方程，而非传递函数，描述非零初始状态的系统，这一点也和连续时间系统相同。卷积和运算与离散时间系统仿真类似，也可以用 filter 实现，而且有多种实现方法。离散时间系统特有的解卷积是卷积的逆运算，是“现代信号处理”等高级信号处理课程的核心内容。

练习题

1. 已知某系统差分方程为

$$y(n)+0.5y(n-1)-0.2y(n-2)-0.1y(n-3)=x(n)-0.3x(n-1)$$

试求其冲激响应和阶跃响应，并判断该系统是否稳定。

2. 若用 $x(n)=0.5^n$ 激励练习题 1 系统，求输出 $y(n)$。

3. 已知某系统结构如图 7.9 所示。当 $n<10$ 时，激励序列 $x(n)=u(n)$，参数 $c=0.1$；当 $n\geqslant 10$ 时，$x(n)=-u(n)$，$c=-0.5$，试求输出 $y(n)$，$0\leqslant n\leqslant 20$。（注意保持滤波器状态连续。）

4. 已知某系统阶跃响应为 $g(n)=n^2u(n)$，试求其冲激响应 $h(n)$。

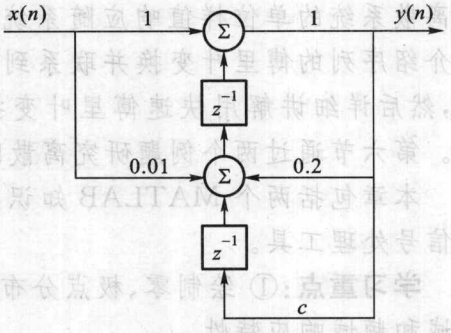


图 7.9　练习题 3 图

第八章　z 变换、离散时间系统的 z 域分析

本章首先介绍 z 变换和逆 z 变换的符号求解方法以及逆 z 变换的部分分式展开法，然后讲解求差分方程的解析解的方法。第四节图示离散系统的单位样值响应随系统函数极点位置改变的变化。第五节介绍序列的傅里叶变换并联系到离散傅里叶变换及其快速实现算法，然后详细讲解用快速傅里叶变换计算连续时间傅里叶变换的步骤。第六节通过两个例题研究离散时间系统的频率响应特性。

本章包括两个 MATLAB 知识点。① 命令和行的关系；② 交互式信号处理工具。

学习重点：① 绘制零、极点分布图；② 由零、极点分布图理解系统时域和频域响应特性。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
ztrans	符号 z 变换	iztrans	符号逆 z 变换
residuez	离散部分分式展开	tf2zp	传递函数模型转为零、极点模型
zp2tf	零、极点模型转为传递函数模型	zplane	绘制离散系统零、极点分布图
fft	快速傅里叶变换	ifft	快速傅里叶逆变换
freqz	计算频率响应		

第一节　z 变换定义、典型序列的 z 变换

MATLAB 中进行 z 变换一般采用符号函数的方法，具体用 ztrans 函数实现[①]。

```
syms n;                          %定义 n 为符号
```

① 和拉氏变换相似，该函数实现的是单边 z 变换。

```
x1 = (1/2)^n;                    %定义 x1
X1 = ztrans(x1)                  %计算 x1 的 z 变换 X1
x2 = n * (n - 1)/2;              %定义 x2
X2 = ztrans(x2)                  %计算 x2 的 z 变换 X2
X2s = simplify(X2)               %将 X2 公式进行简化得到 X2s
```

计算结果如下

```
X1 =
2 * z/(2 * z - 1)
X2 =
1/2 * z * (z + 1)/(z - 1)^3 - 1/2 * z/(z - 1)^2
X2s =
z/(z - 1)^3
```

即 $X_1=\dfrac{2z}{2z-1}$，$X_2=\dfrac{1}{2}\dfrac{z(z+1)}{(z-1)^3}-\dfrac{1}{2}\dfrac{z}{(z-1)^2}$，$X_{2s}=\dfrac{z}{(z-1)^3}$。

请读者自行练习做其他典型序列的 z 变换，并用课本上结果进行验证。注意：尽管 heaviside 函数明显针对 $u(t)$，而非 $u(n)$ 定义，因为它将零时刻定义为 NaN 而不是 1。但调用

```
ztrans(heaviside(n));
```

仍然可以得到正确的结果

```
ans =
z/(z - 1)
```

所以读者也可以将 heaviside 作为离散阶跃序列使用。

第二节　逆 z 变换

同 z 变换相似，可以用符号运算函数 iztrans 实现逆 z 变换[①]。建议读者选用主教材表 8-2 中的若干变换式进行验证。

为了加深对概念的理解，和连续时间系统做部分分式展开的 residue 函数一样，推荐用部分分式展开法实现逆 z 变换，即 residuez 函数。用该函数可以得到对应于各个部分分式的极点和系数，进而查表导出序列。residuez 函数的调用方法和 residue 非常相似，因而不再赘述，直接给出例题演示。

例 8.1(主教材例 8-5)　用部分分式展开法求解

① 同样只能求出右边序列，即收敛域在圆的外侧。

$$X(z)=\frac{z^2}{z^2-1.5z+0.5}$$

的逆变换 $x(n)(|z|>1)$。

解 先用符号方法求逆 z 变换。在命令窗口中输入

```
syms z;                          %定义符号 z
X = z^2/(z^2 - 1.5 * z + 0.5);   %定义变换式 X
x = iztrans(X)                   %计算 X 的逆变换 x
x =
 -(1/2)^n + 2
```

符号法算得 $x(n)=-\left(\frac{1}{2}\right)^n+2$。接下来用部分分式展开法求解。为了准确写出系数 a 和 b，首先将原式写成标准形式

$$X(z)=\frac{1}{1-1.5z^{-1}+0.5z^{-2}}$$

```
a = [1, -1.5,0.5];               %定义多项式分母的系数 a
b = 1;                           %定义多项式分子的系数 b
[r,p,k] = residuez(b,a)          %部分分式展开，得到常数项 r，极点 p
                                 %和直接项 k
r =                              %两个常数项
2
 -1
p =                              %两个极点
1.0000
0.5000
k =                              %没有直接项，因为分子比分母阶数低
[]
```

根据 residuez 的输出，可直接写出部分分式展开式为

$$X(z)=\frac{2}{1-z^{-1}}-\frac{1}{1-0.5z^{-1}}$$

再由主教材表 8-2 查得

$$x(n)=(2-0.5^n)u(n)$$

可见部分分式展开法得到的结果和符号法得到的结果完全相同[①]。

① 注意由于逆 z 变换输出为右边序列，所以结果等同于乘以 $u(n)$。

和 residue 函数类似，当有重根、共轭根或者分子阶数高于分母时，residuez 的输出比较复杂，请选择适当的题目对照 residue 学习和实验。

第三节　利用 z 变换解差分方程

在 MATLAB 环境下，求解差分方程从时域用迭代法最为方便，但这样只能得到数值结果；若需要闭式解，只能用符号运算。下面分零状态响应和完全响应两种情况举例介绍。

求零状态响应，可以用 ztrans 求出传递函数和激励序列的 z 变换式，相乘后再用 iztrans 得到输出的表达式。

例 8.2(主教材例 8-16)　一离散系统的差分方程为

$$y(n)-by(n-1)=x(n)$$

若激励 $x(n)=a^nu(n)$，起始值 $y(-1)=0$，求响应 $y(n)$。

解　容易看出响应为右边序列，因而可以用符号法求解逆 z 变换。

§§§MATLAB 文件 ex_8_2.m

```
syms n a b z                    %定义符号 n,a,b,z
x = a^n;                        %定义激励信号 x(n)
X = ztrans(x);                  %计算激励信号的变换 X(z)
H = 1/(1 - b * z^(-1));         %由差分方程直接写出系统 z 变换式 H(z)
Y = H * X;                      %计算输出的变换式 Y(z)=H(z)X(z)
y1 = iztrans(Y);                %计算输出时域表达式 y1(n)
y = simplify(y1)                %将 y1(n)简化成 y(n)
```

输出结果为

```
y =
(a^(1 + n) - b^(1 + n))/(a - b)
```

即 $y(n)=\dfrac{a^{n+1}-b^{n+1}}{a-b}u(n)$，注意要在结果上乘以 $u(n)$。上例中 simplify 函数用于化简多项式。

对起始值不为零的完全响应，只能利用位移性质手工计算出输出序列的 z 变换式，然后用 iztrans 得到时域表达式。

例 8.3(主教材例 8-17)　对于上例的差分方程，若激励不变，但起始值不等于零，而是 $y(-1)=2$，求系统的响应 $y(n)$。

解　对差分方程两边取单边 z 变换，由位移性质得到

$$Y(z)-bz^{-1}Y(z)-by(-1)=X(z)$$

代入 $y(-1)=2$ 得到

$$Y(z)=\frac{X(z)+2b}{1-bz^{-1}}$$

```
§§§MATLAB 文件 ex_8_3.m
syms a b z;                          %定义符号 a,b,z
X = z/(z-a);                         %定义激励的 z 变换 X(z)
Y = (X+2*b)/(1-b*z^(-1));            %由推导得到的公式计算 Y(z)
y1 = iztrans(Y);                     %计算 Y(z)的逆变换 y1(n)
y = simplify(y1);                    %简化 y1(n)为 y(n)
```

运行结果为

```
y =
(a^(1+n) - b^(1+n) + 2*b^(1+n)*a - 2*b^(2+n))/(a-b)
```

即 $y(n)=\dfrac{a^{n+1}-b^{n+1}+2b^{n+1}a-2b^{n+2}}{a-b}=\dfrac{a^{n+1}-b^{n+1}}{a-b}+2b^{n+1}$。

第四节 离散系统的系统函数

单位样值响应和系统函数是 z 变换对。MATLAB 中，若用符号方法，由 ztrans 和 iztrans 可以在时域及变换域的表达式之间互求；若用数值方法，可以用 impz 由系统函数分子分母的多项式系数求单位样值响应。

同连续时间系统一样，MATLAB 提供了在传递函数（多项式）和零、极点模型之间转换的 tf2zp 和 zp2tf 函数，从而便于研究零、极点分布对系统函数进而对单位样值响应的影响。但根据 MATLAB Help，对离散系统使用 tf2zp 有较苛刻的条件：建议用 eqtflength 函数将传递函数的分子和分母多项式系数长度调整到相同，然后再调用 tf2zp 计算零、极点。

在 MATLAB 中，为考察系统的稳定性，可以直观地用 zplane 绘制零、极点分布图，如果有极点位于单位圆外，则系统不稳定。

例 8.4（主教材图 8-12 修改） 当 $H(s)$ 极点（一阶）位于图 8.1 所示 z 平面中各方框附近的（极点）位置

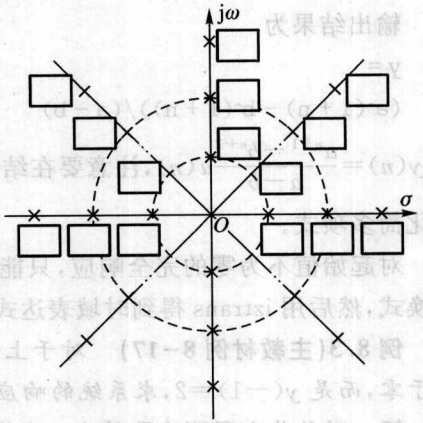


图 8.1 例 8.4 极点位置图

时，画出对应的 $h(n)$ 波形填入方框中。

解　§§§MATLAB 文件 ex_8_4.m

```
pos = [26,19,18,17,24,27,...        %按顺序作图的子图 id
13,11,9,23,28,7,4,1,22];
figure,id = 1;                      %生成新图框，并将子图 id 初始化为 1
for r = 0.8:0.2:1.2                 %循环：极点的幅度依次为 0.8,1.0,1.2
  for theta = 0:pi/4:pi             %循环：极点的弧角依次为 0,π/4,π/2
    p = r * exp(j * theta);         %生成极点
  if theta~ = 0 & theta~ = pi
                                    %如果极点不在实轴上，添加一个共轭
                                    %极点
    p = [p;p'];
  end
  [b a] = zp2tf([],p,1);            %由零、极点得到传递函数，注意无零点
                                    %且 k=1
  subplot(4,7,pos(id));             %选定画图的子图位置
  [h,t] = impz(b,a,20);             %计算 20 个点的单位样值响应
```

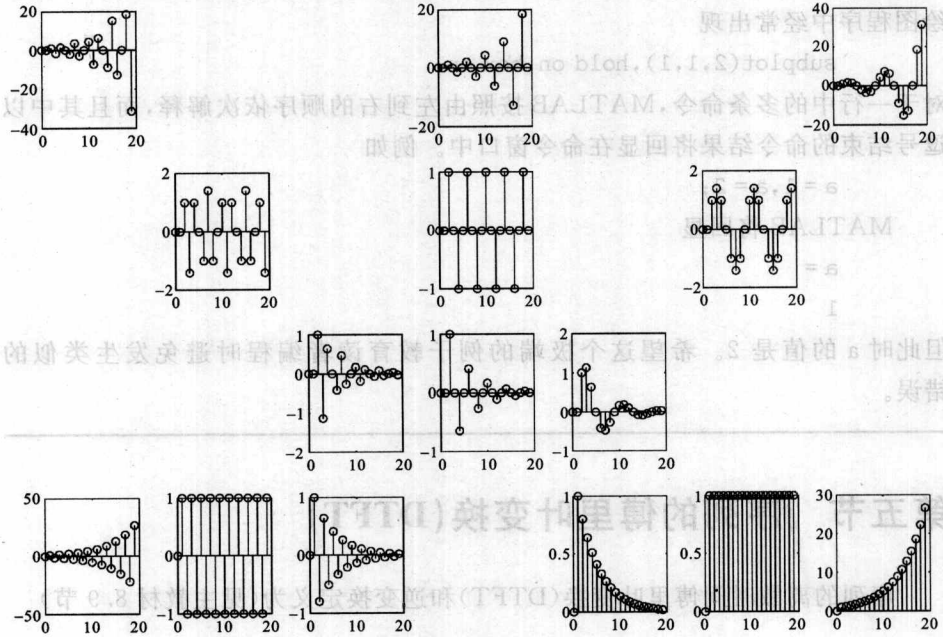


图 8.2　例 8.4 运行结果

```
        stem(t,h,'k-','MarkerSize',5);          %绘制单位样值响应
        id = id + 1;                            %子图序号加 1
    end                                         %退出弧角循环
end                                             %退出幅度循环
```

运行结果如图 8.2 所示。zplane 函数将在本章第六节中举例介绍。

MATLAB 知识点(14)——命令和行的关系

MATLAB 定义矩阵、结构和单元数组时用分号“;”或回车分开各行,因而矩阵定义命令可能占据多行。但一般而言,每条 MATLAB 命令只占一行。如果有命令太长一行容纳不下,或者有其他原因希望将一条命令显示在多行,可以用符号“...”表示续行,例如

```
set(gca,...
    'XLim',[0 1],'YLim',[10,1000],...
    'XScale','linear','YScale','log');
```

另一方面,一般每行只写一条命令,这样做程序结构清晰,便于阅读和理解。但有时将功能相似的短命令写在一行显得更加紧凑,此时可用逗号“,”或分号“;”将同一行中的多个命令分开,正如读者在中国高校电工电子课程网上看到的绘图程序中经常出现

```
subplot(2,1,1),hold on,box on;
```

对于一行中的多条命令,MATLAB 按照由左到右的顺序依次解释,而且其中以逗号结束的命令结果将回显在命令窗口中。例如

```
a = 1,a = 2;
```

MATLAB 将回显

```
a =
     1
```

但此时 a 的值是 2。希望这个极端的例子教育读者编程时避免发生类似的错误。

第五节 序列的傅里叶变换(DTFT)

序列的离散时间傅里叶变换(DTFT)和逆变换定义为(见主教材 8.9 节)

$$\mathrm{DTFT}[x(n)] = X(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{n=-\infty}^{\infty} x(n)\mathrm{e}^{-\mathrm{j}\omega n} \tag{8.1}$$

$$\mathrm{IDTFT}[X(\mathrm{e}^{\mathrm{j}\omega})]=x(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi}X(\mathrm{e}^{\mathrm{j}\omega})\mathrm{e}^{\mathrm{j}\omega n}\,\mathrm{d}\omega \tag{8.2}$$

和傅里叶变换、拉氏变换及 z 变换不同，MATLAB 没有直接提供计算 DTFT 的符号函数方法。但可以先用 ztrans 计算出 z 变换，然后将 $z=\mathrm{e}^{\mathrm{j}\omega}$ 代入，即可得到 DTFT 变换。

若序列 $x(n)$ 是有限长的，即当且仅当 $0\leqslant n<N$ 时 $x(n)\neq 0$，就可以用数值方法计算 DTFT 中 $[-\pi,\pi]$ 区间内 N 个抽样点的值，$X(k)=X(\mathrm{e}^{\mathrm{j}k\frac{2\pi}{N}})$。事实上，从时域序列到这些频域抽样点的过程被赋予了一个专用名称——离散傅里叶变换(DFT)。

$$\mathrm{DFT}[x(n)]=X(k)=\sum_{n=0}^{N-1}x(n)\mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{N}} \tag{8.3}$$

$$\mathrm{IDFT}[X(k)]=x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)\mathrm{e}^{\mathrm{j}nk\frac{2\pi}{N}} \tag{8.4}$$

为了实现 DFT，在 MATLAB 中可以用下述公式描述

$$\boldsymbol{X}=\boldsymbol{W}\boldsymbol{x} \tag{8.5}$$

$$\boldsymbol{x}=\frac{1}{N}\boldsymbol{W}^{\mathrm{H}}\boldsymbol{X} \tag{8.6}$$

其中 $\boldsymbol{x}=[x(0),x(1),\cdots,x(N-1)]^{\mathrm{T}}$，$\boldsymbol{X}=[X(0),X(1),\cdots,X(N-1)]^{\mathrm{T}}$

$$\boldsymbol{W}=\begin{bmatrix} w^0 & w^0 & w^0 & \cdots & w^0 \\ w^0 & w^1 & w^2 & \cdots & w^{N-1} \\ w^0 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ w^0 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{bmatrix} \tag{8.7}$$

其中 $w=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$，回想前面介绍的用数值方法求连续时间的傅里叶级数和傅里叶变换，都存在一定的近似，而上述公式准确地实现了离散傅里叶变换，因而有 $\boldsymbol{W}^{\mathrm{H}}\boldsymbol{W}=N\boldsymbol{I}$。

离散傅里叶变换在主教材第九章 9.3 节讲到，也是“信号与系统”后继课程“数字信号处理”的重要内容。DFT 的计算复杂度是 N^2 量级，随着 N 的增加计算量迅速增大，严重降低了 DFT 的实用性。但由于矩阵 $\boldsymbol{W}$ 存在周期性和对称性，因而出现了多种低复杂度($N\log_2 N$ 量级)的简化算法，这些算法统称为快速傅里叶变换(FFT)(主教材第九章 9.6 节)。MATLAB 提供了 fft 和 ifft 函数分别实现式(8.3)和式(8.4)，下面将重做例 4.2，向读者介绍用 fft 求连续时间傅里叶变换的方法。

将第四章第一节中给出的傅里叶变换和逆变换的求和近似式重写如下

$$F(\omega_1+k\Delta\omega)=\frac{T}{N}\sum_{n=0}^{N-1}f(t_1+n\Delta t)\mathrm{e}^{-\mathrm{j}(\omega_1+k\Delta\omega)(t_1+n\Delta t)} \tag{8.8}$$

$$f(t_1+n\Delta t)=\frac{\Omega}{2\pi K}\sum_{k=0}^{K-1}F(\omega_1+k\Delta\omega)\mathrm{e}^{\mathrm{j}(\omega_1+k\Delta\omega)(t_1+n\Delta t)} \tag{8.9}$$

其中$[t_1,t_2]$为时域主要取值区间，$T=t_2-t_1$ 为信号近似长度，$\Delta t=\dfrac{T}{N}$为抽样间隔，N 为时域抽样点数；$[\omega_1,\omega_2]$为信号主要频带，$\Omega=\omega_2-\omega_1$ 为近似带宽，$\Delta\omega=\dfrac{\Omega}{K}$为频域抽样间隔，$K$ 为频域抽样点数。

为了使第四章中定义的变换矩阵 $\boldsymbol{U}$ 和逆变换矩阵 $\boldsymbol{V}$ 从形式上向 DFT 的 $\boldsymbol{W}$ 上靠拢，定义两个新的变量 $f_1(t)$和 $F_1(\omega)$

$$f_1(t)=f(t)\mathrm{e}^{-\mathrm{j}\omega_1 t}$$

$$F_1(\omega)=F(\omega)\mathrm{e}^{\mathrm{j}\omega t_1}$$

并用它们替换掉式(8.8)中的 $f(t)$和 $F(\omega)$，化简后得到

$$F_1(\omega_1+k\Delta\omega)\approx\frac{T\mathrm{e}^{\mathrm{j}\omega_1 t_1}}{N}\sum_{n=0}^{N-1}f_1(t_1+n\Delta t)\mathrm{e}^{-\mathrm{j}k\Delta\omega n\Delta t} \tag{8.10}$$

将上式写成矩阵形式，有

$$\boldsymbol{F}_1=\frac{T\mathrm{e}^{\mathrm{j}\omega_1 t_1}}{N}\boldsymbol{U}\boldsymbol{f}_1 \tag{8.11}$$

其中

$$\boldsymbol{f}_1=[f_1(t_1),f_1(t_1+\Delta t),\cdots,f_1(t_2-\Delta t)]^{\mathrm{T}}$$

$$\boldsymbol{F}_1=[F_1(\omega_1),F_1(\omega_1+\Delta\omega),\cdots,F_1(\omega_2-\Delta\omega)]^{\mathrm{T}}$$

$$\boldsymbol{U}=\begin{bmatrix}u^0 & u^0 & u^0 & \cdots & u^0\\ u^0 & u^1 & u^2 & \cdots & u^{N-1}\\ u^0 & u^2 & u^4 & \cdots & u^{2(N-1)}\\ \vdots & \vdots & \vdots & & \vdots\\ u^0 & u^{K-1} & u^{2(K-1)} & \cdots & u^{(K-1)(N-1)}\end{bmatrix} \tag{8.12}$$

其中 $u=\mathrm{e}^{-\mathrm{j}\Delta\omega\Delta t}$。对比 $\boldsymbol{U}$ 和 $\boldsymbol{W}$ 的表达式，可见若 $K=N$，则其形式完全相同；如果进一步 $u=w$，则有 $\boldsymbol{U}=\boldsymbol{W}$。

对于傅里叶逆变换，同样有

$$\boldsymbol{f}_1=\frac{\Omega \mathrm{e}^{-\mathrm{j}\omega_1 t_1}}{2\pi K}\boldsymbol{V}\boldsymbol{F}_1 \tag{8.13}$$

$$\boldsymbol{V}=\begin{bmatrix} u^0 & u^0 & u^0 & \cdots & u^0 \\ u^0 & u^{-1} & u^{-2} & \cdots & u^{-(K-1)} \\ u^0 & u^{-2} & u^{-4} & \cdots & u^{-2(K-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ u^0 & u^{-(N-1)} & u^{-2(N-1)} & \cdots & u^{-(K-1)(N-1)} \end{bmatrix}=\boldsymbol{U}^{\mathrm{H}} \tag{8.14}$$

在 $K=N$ 和 $u=w$ 的约束下，得到

$$\Omega T=2\pi N \tag{8.15}$$

即只要选择 Ω，T 和 N 满足以上条件，即可调用 fft 函数代替矩阵乘法近似计算连续时间的傅里叶变换。因而有

$$\boldsymbol{F}_1=\frac{T\mathrm{e}^{\mathrm{j}\omega_1 t_1}}{N}\mathrm{fft}(\boldsymbol{f}_1) \tag{8.16}$$

$$\boldsymbol{f}_1=\frac{\Omega \mathrm{e}^{-\mathrm{j}\omega_1 t_1}}{2\pi}\mathrm{ifft}(\boldsymbol{F}_1) \tag{8.17}$$

例 8.5(主教材图 3-21 修改)　请绘制矩形脉冲

$$f(t)=\begin{cases}1 & |t|<\dfrac{1}{2} \\ 0 & \text{其他}\end{cases}$$

的波形($t\in[-1,1]$)和频谱。

解　题目给定 $T=2$，计划用 100 个抽样点表示波形和频谱，即 $N=K=100$，代入式(8.15)可确定出 $\Omega=100\pi$。下面分别用 fft 函数和左乘变换矩阵的方法计算频谱，然后再分别用 ifft 函数和左乘逆变换矩阵恢复时域信号，并验证两者完全相同。

§§§MATLAB 文件 ex_8_5.m

```
T = 2;                                          %定义信号抽样区间
                                                %长度
N = 100;                                        %定义时域和频域抽
                                                %样点数
OMG = 100 * pi;                                 %定义频域抽样区间
[t,omg,FT,IFT] = prefourier([ - T/2,T/2],N,[ - OMG/2,OMG/2],N);
f = 0 * t;                                      %初始化时域波形
f(t> - 1/2 &t<1/2) = 1;
```

```
F = FT * f;                                    %用矩阵做傅里叶变换
fs = IFT * F;                                  %用矩阵做傅里叶逆
                                               %变换

f1 = f. * exp( - j * omg(1) * t);              %生成辅助时域函数 f1
F1 = T * exp(j * omg(1) * t(1))/N * fft(f1);   %用 fft 函数得到 F1
F_fft = F1. * exp(j * omg * t(1));             %从 F1 恢复到频谱
                                               %F_fft

f1_ifft = OMG * exp( - j * omg(1) * t(1))/2/pi * ifft(F1);
                                               %利用 ifft 函数做逆
                                               %变换
f_ifft = f1_ifft. * exp(j * omg(1) * t);       %由得到的时域辅助
                                               %信号恢复 f_ifft

ex_8_5_fft_plot();                             %绘制输出波形和频谱
```

两种方法计算出的频谱和恢复出的波形如图 8.3 所示。可见两种方法得到的频谱和时域波形完全相同。

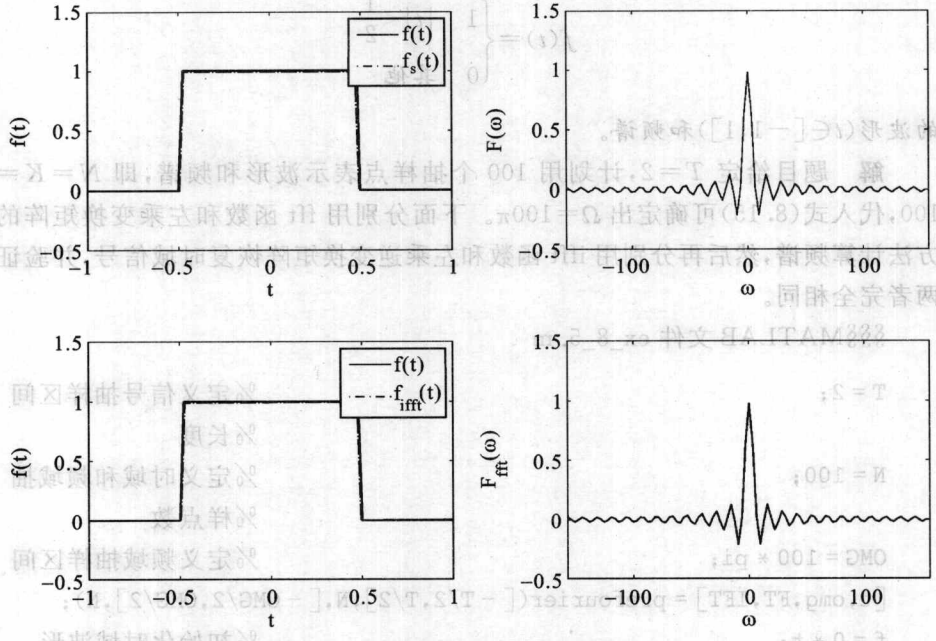


图 8.3 例 8.5 信号波形和频谱

第六节　离散时间系统的频率响应特性

MATLAB提供了freqz函数由系统函数绘制频率响应。下例将观察零、极点分布图，单位样值响应和频响特性。

例 8.6(主教材例 8-23 修改)　求图8.4所示二阶离散系统的频率响应，其中$a_1=1.1, a_2=-0.7, b_1=1$。

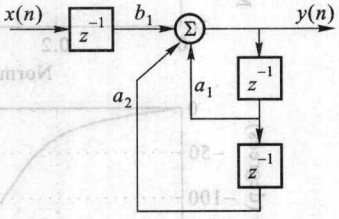


图 8.4　例 8.6 系统框图

解　该系统差分方程为

$$y(n)-a_1y(n-1)-a_2y(n-2)=b_1x(n-1)$$

§§§MATLAB文件 ex_8_6.m

```
a1 = 1.1,a2 = -0.7,b1 = 1;          %为参数赋值
a = [1, -a1, -a2];                  %定义左侧系数
b = [0,b1];                         %定义右侧系数
figure;                             %生成新图框
subplot(2,1,1),zplane(b,a);         %在第一个子图内绘制零、极点分布图
subplot(2,1,2),impz(b,a);           %在第二个子图内绘制单位样值响应
figure,freqz(b,a);                  %生成新图框并绘制频率特性
```

运行结果如图8.5和图8.6所示。可见该系统表现为一个带通滤波器。

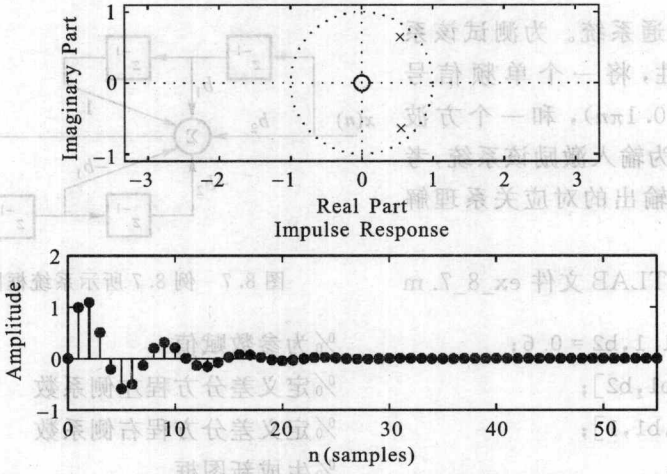


图 8.5　例 8.6 所示系统的零、极点分布图和单位样值响应

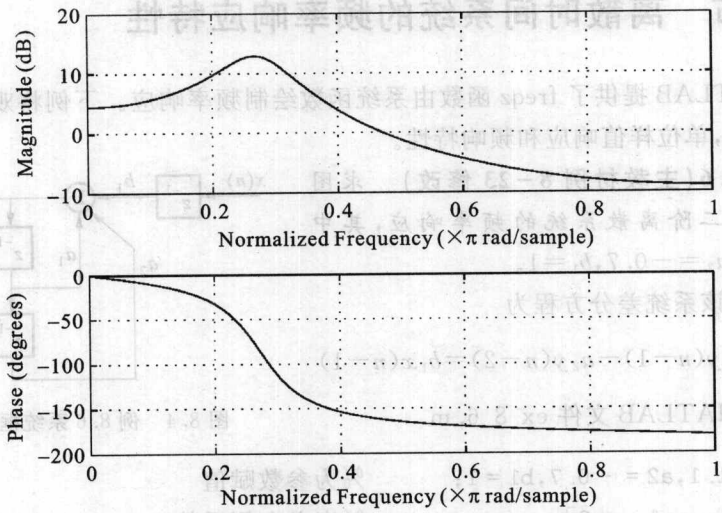


图 8.6 例 8.6 所示系统的幅频响应和相频响应

例 8.7(主教材例 8-24 修改) 求图 8.7 所示离散系统的频率响应,其中 $b_1=-1.1, b_2=0.6$。已知该系统为全通系统,请设计输入信号并验证该系统性能。

解 由该系统的差分方程

$$y(n)+b_1y(n-1)+b_2y(n-2)=b_2x(n)+b_1x(n-1)+x(n-2)$$

可知其为全通系统。为测试该系统频响特性,将一个单频信号 $x_1(n)=\sin(0.1\pi n)$,和一个方波信号分别作为输入激励该系统,考察其输入和输出的对应关系理解全通系统。

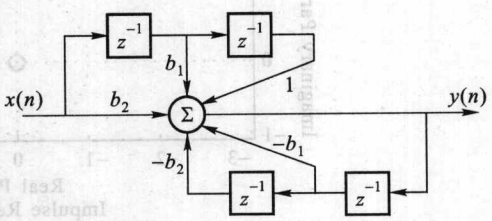


图 8.7 例 8.7 所示系统框图

§§§MATLAB 文件 ex_8_7.m

```
b1 = -1.1,b2 = 0.6;                  %为参数赋值
a = [1,b1,b2];                       %定义差分方程左侧系数
b = [b2,b1,1];                       %定义差分方程右侧系数
figure;                              %生成新图框
subplot(2,1,1),zplane(b,a);          %在第一个子图内绘制零、极点分布图
subplot(2,1,2),impz(b,a);            %在第二个子图内绘制单位样值响应
```

```
figure,freqz(b,a);                    %生成新图框并绘制频率响应
n = [0:40]';                          %生成时间点
x1 = sin(0.1 * pi * n);               %生成单频信号
x2 = 0 * n;                           %准备方波信号
x2(mod(n,10)<5) = 1;                  %生成方波信号,周期是 10
y1 = filter(b,a,x1);                  %分别对两个信号滤波
y2 = filter(b,a,x2);
ex_8_7_plot();                        %绘制输出图形
```

上述全通系统的零、极点分布和单位样值响应如图 8.8 所示。可见其极点和零点位置关于单位圆对称。幅频响应和相频响应如图 8.9 所示。可见该系统对所有频率分量的增益都为 1，这也是称为全通系统的由来；但是相位延时随频率变化(注意非线性变化)。

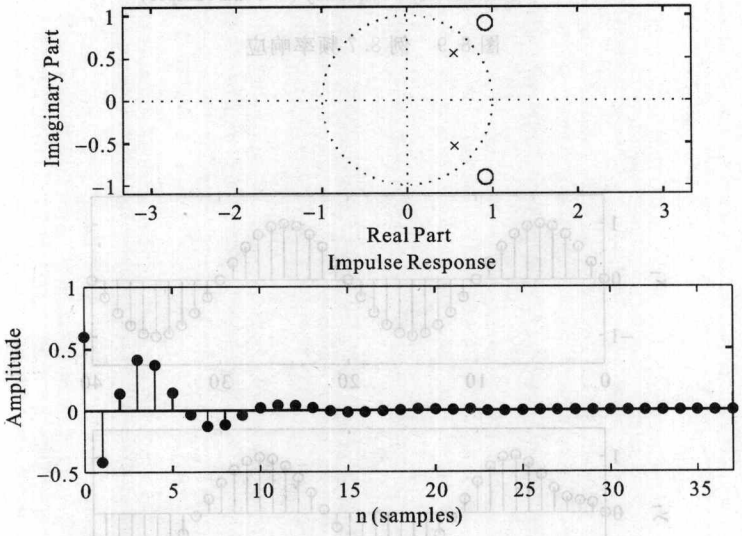


图 8.8　例 8.7 零、极点分布图和单位样值响应

两个输入信号及其响应如图 8.10 和图 8.11 所示。可见任意频率的单频信号经过全通系统后出现相位延迟，但幅度不发生变化。宽带信号(方波中有多个频率分量)经过全通滤波器后除了有相位延迟外，波形剧烈变化。根据频响曲线，虽然对各个频率分量其幅度增益都是 1，但相位延迟非线性变化，因而输出波形变得面目全非。

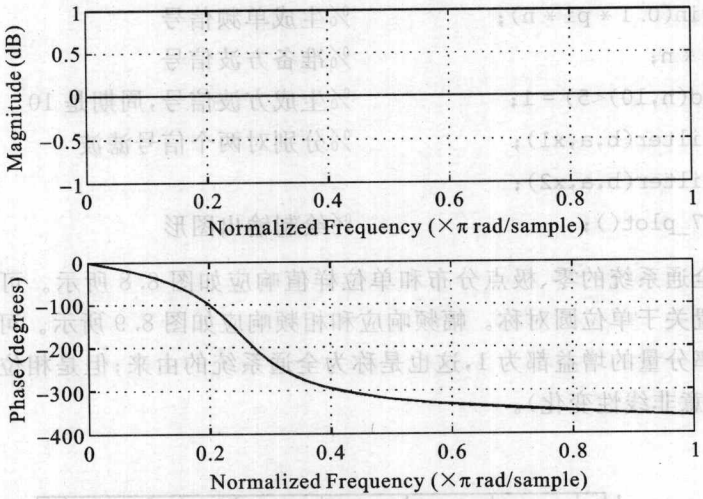


图 8.9 例 8.7 频率响应

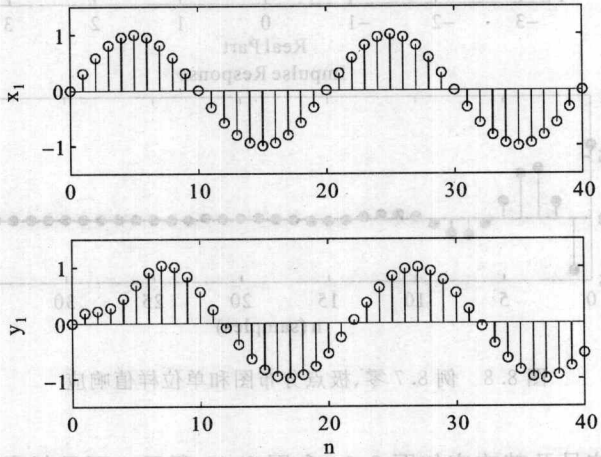


图 8.10 例 8.7 单频输入信号及其响应

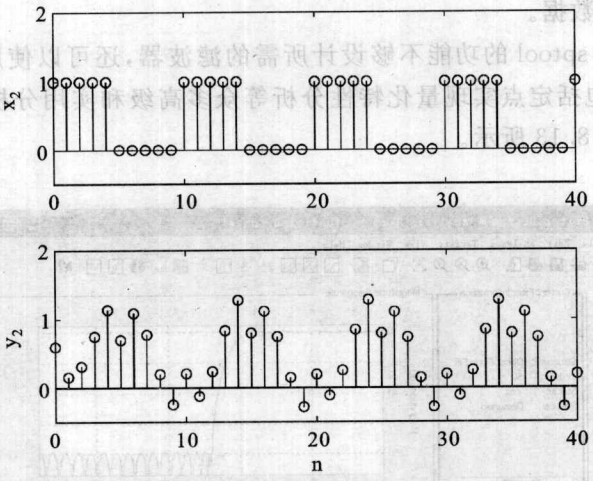


图 8.11　例 8.7 方波输入信号及其响应

MATLAB 知识点(15)——交互式信号处理工具

图形用户界面的信号处理工具 sptool 下集成了其他四个图形界面工具，分别用于观察信号(Signal Browser)、设计滤波器(Filter Designer)、滤波器可视化(Filter Visualization Tool)和观察频谱(Spectrum Viewer)。这几个工具中又集成了多种信号处理、滤波器分析和谱分析的函数。

Signal Browser 用于观察、测量和分析信号的各种时域信息。

Filter Designer 用于设计和编辑不同长度和类型的 FIR 或 IIR 滤波器，支持包括低通、高通、带通、带阻和多带通等标准需求。

Filter Visualization Tool 即 fvtool，可用来查看滤波器的幅度响应，相位响应，群延时，相位延时，零、极点分布图，冲激响应和阶跃响应等。fvtool 图形界面如图 8.12 所示。

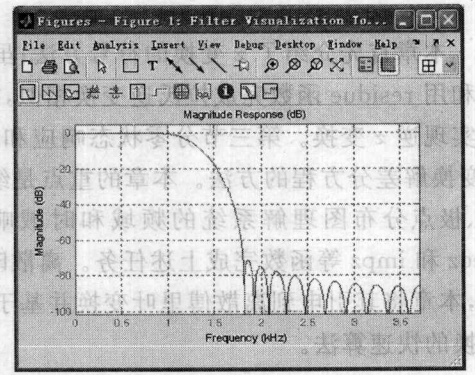


图 8.12　fvtool 图形界面

Spectrum Viewer 支持包括 Burg、FFT、multitaper、MUSIC 特征矢量、Welch 和 Yule-Walker 自回归在内的多种谱密度估计方法，

用于分析频响数据。

如果觉得 sptool 的功能不够设计所需的滤波器，还可以使用 fdatool，它功能更丰富，它包括定点实现量化特性分析等众多高级和实用分析工具。fdatool 图形界面如图 8.13 所示。

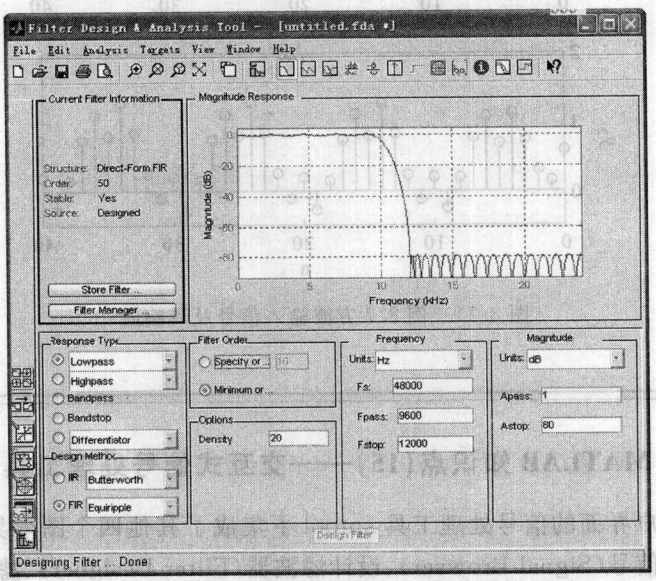


图 8.13 fdatool 图形界面

第七节 小结

本章首先介绍了 z 变换的符号方法；虽然也可以用符号方法求解逆 z 变换，但和用 residue 函数完成拉氏逆变换相似，推荐用部分分式展开法的 residuez 函数实现逆 z 变换。第三节分零状态响应和完全响应两种情况，举例介绍了利用 z 变换解差分方程的方法。本章的重点是绘制系统的零、极点分布图，以及通过零、极点分布图理解系统的频域和时域响应特性，MATLAB 提供了 zplane、freqz 和 impz 等函数完成上述任务。离散时间傅里叶变换就是单位圆上的 z 变换，本章将其引申到离散傅里叶变换并基于 fft 函数介绍了计算连续时间傅里叶变换的快速算法。

练习题

1.（主教材习题 8-12 修改）画出 $X(z)=\dfrac{-3z^{-1}}{2-5z^{-1}+2z^{-2}}$ 的零、极点分布图。计算并绘制收敛域为 $|z|>2$ 情况下的对应序列。

2.（主教材习题 8-24 修改）已知某因果离散系统的差分方程为

$$y(n)+3y(n-1)-y(n-2)=x(n)$$

试求：(1) 系统的单位样值响应 $h(n)$；

(2) 若 $x(n)=(n+n^2)[u(n)-u(n-6)]$，求响应 $y(n)$。

3. 已知某离散时间系统极点为 $p=\{0.1,0.5\pm j0.2,-0.9\}$，零点为 $z=\{0,1\}$，绘制该系统的单位样值响应、单位阶跃响应和幅频、相频响应。

第九章　语音合成

本章将基于数字滤波器和 z 变换等基础知识，应用第一篇讲授的 MATLAB 编程技术，在语音分析合成领域做一些练习。通过本章的练习，可以增进对 z 变换和滤波器的理解，熟练运用 MATLAB 基本指令。本章包括两部分，第一部分介绍语音生成和分析的基本知识，第二部分给出详细的练习内容和编程步骤。相信读者对此也会产生强烈兴趣。

第一节　背景知识

9.1.1　发声机理

从物理原理来看，语音信号是由肺挤压出的空气激励发声器官带来的震动产生的。发声器官包括喉、声道和嘴。喉位于气管的上端，实际上是气管末端的一圈软骨构成的一个框架。喉中有两片肌肉，称为声带。声带张开时空气可以自由地流过喉和气管，如正常呼吸动作；声带闭合时将喉封住，所以吃东西时食物不会落入气管。两片声带之间的空隙称为声门。说话时声带相互靠拢但不完全封闭，这样声门变成一条窄缝，当气流通过时压力减小，从而声带完全合拢使气流不能通过；在气流被阻断时压力恢复正常，因而声带间的空隙形成，气流再次通过。这一过程周而复始，就形成了一串周期性的脉冲气流送入声道，如图 9.1 所示。这个脉冲串的周期称为“基音周期”，其倒数是“基音频率”。男性说话的基音频率在 60～200 Hz 范围内，女性和小孩在 200～450 Hz 之间。这种方式发出的音就是浊音。

气流从喉向上经过口腔或者鼻腔后向外辐射，其间的传输通道称为声道。气流流过声道时犹如通过一个具有某种谐振特性的腔体，称为声管，如图 9.2 所示。输出气流的频率特性既取决于声门脉冲串的特性，又取决于声道特性。声

图 9.1　典型的声门脉冲串波形

道包括口腔和鼻腔两部分，对成年男性而言，口腔段约 17 cm，鼻腔段约 13 cm，气流在软腭的控制下分别流向这两个通道。声道的截面积并非常数，而声道的频率特性主要取决于声道截面积的最小值（收紧点）出现的位置，除了软腭控制一些外，收紧点主要由舌头的位置来决定。

图 9.2　声管构造示意图

语音的另一种产生方式是声门完全闭合，此时声道不是受声门周期脉冲气流的激励，而是利用口腔内存有的空气释放出来而发声。该气流在口腔中形成湍流，因而带有明显的随机噪声的特点。这种方式发出的音就是清音。（如果把手放在脖子前面喉结上部的倒三角位置，发浊音“啊”的音时可以感觉到振动，发清音“科”时就感觉不到。）

9.1.2　语音信号的时域特征

一段女声发音“MATLAB”的波形如图 9.3 所示，可以看出语音能量的起伏以及大致分辨出话语中的每个音节在此波形中的位置。把时间轴拉宽后在图 9.4 中观察两个细节部分，可以看出语音的浊音段能量较大（上图），有明显的周期特征，而清音段能量很小（下图），类似于噪声随机变化。

9.1.3　语音模型

语音生成模型

通过对声管的研究，发现它可以用若干段级连的不等截面积均匀管道进行描述，如图 9.5 所示，一般称作级连无损声管模型。采用流体力学的方法可以证明每一截均匀管道的频响能够用一个单极点模型来近似，这样 N 段管道组成的

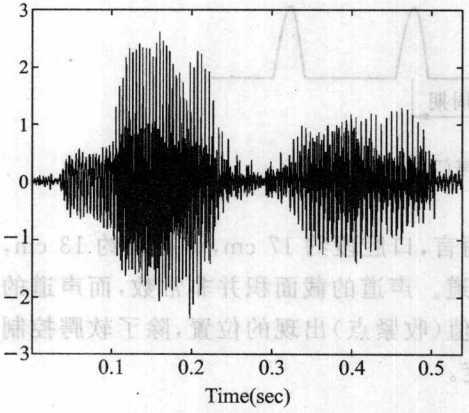


图 9.3 女声发音“MATLAB”

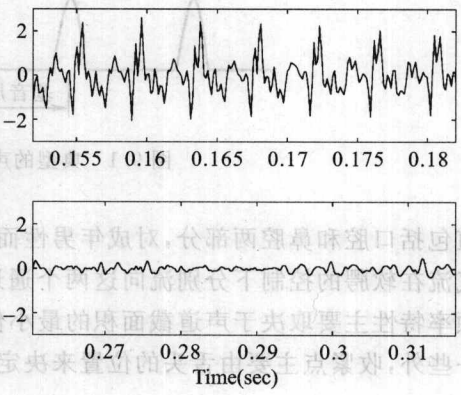


图 9.4 女声发音“MATLAB”细节

声管就可以用一个 N 阶全极点滤波器表述，即

$$V(z)=\frac{G}{\prod_{k=1}^{N}(1-p_k z^{-1})}=\frac{G}{1-\sum_{k=1}^{N}a_k z^{-k}} \tag{9.1}$$

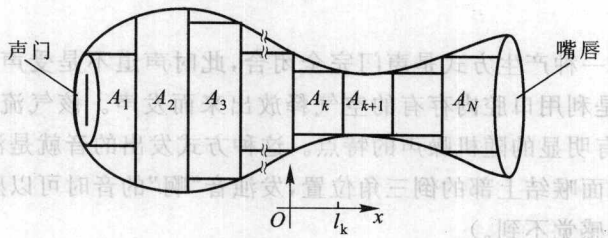


图 9.5 级连无损声管模型

对于典型的男声，$N=10$，所有的极点 p_i 要分别构成共轭对以保证$\{a_i\}$系数都是实数。再综合考虑清音信号，就可以得到产生语音信号的离散语音模型，如图 9.6 所示。

准确的清浊音判决远远超出了本书的范畴，因而将对上述模型进行充分简化。首先去掉随机信号激励部分，认为激励信号是一个脉冲序列，不考虑有无周期。其次去掉声门脉冲模型和口唇的辐射模型，从而得到图 9.7 所示最简单的语音模型，现在只用 z 变换的知识就可以应对了。

假设激励信号用 $e(n)$ 表示，语音信号用 $s(n)$ 表示，根据全极点模型表达式，有

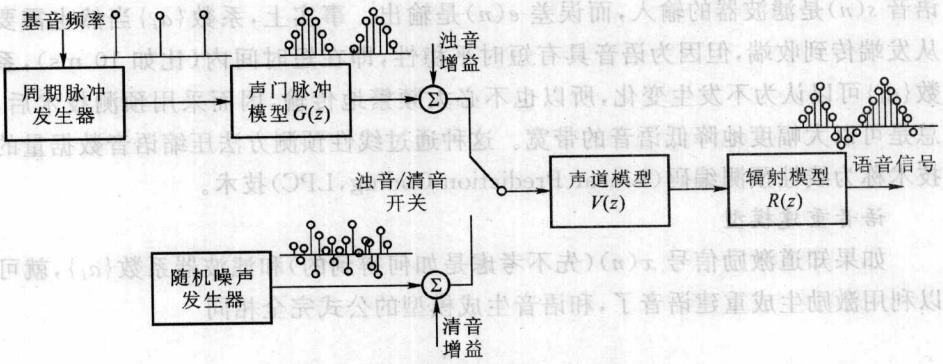


图 9.6　产生语音信号的离散时域模型

图 9.7　简化的语音生成模型

$$s(n)=\sum_{k=1}^{N} a_k s(n-k)+Ge(n) \tag{9.2}$$

从而可以用声管模型对激励信号进行滤波得到语音信号。

语音预测模型

我们可以采集到语音信号 $s(n)$，也已经知道了它的生成模型如图 9.7 所示，但不知道激励 $e(n)$和模型 $V(z)$中的$\{a_i\}$系数。根据主教材第七章 7.7 节，可以知道这是一个解卷积问题，而且它是更复杂的盲解卷，因为激励和滤波器系数两者都不知道。如果进一步做些合理的假设，这个问题还是可以解决的，比如约束 $e(n)$是一个周期脉冲序列和一个高斯白噪声序列之和，就可以用一些信号处理方法，如自相关法和自协方差法求出$\{a_i\}$系数来，并且有 Durbin 递推算法和 Schur 递推算法等快速方法。

假设已经知道了系数$\{a_i\}$，那么将图 9.7 的输入和输出对换，就构成了语音的预测模型，即语音信号 $s(n)$送入预测滤波器，得到预测残差 $e(n)$

$$e(n)=s(n)-\sum_{k=1}^{N} a_k s(n-k)$$

这种预测模型在通信中用来增加每个信道上传输语音信号的通道数。假设信号的发端和收端都知道预测系数$\{a_i\}$，那么发端只需要把误差 $e(n)$传到收端即可，因为收端可以用 $e(n)$作为上述差分方程的激励得到重建语音。在发端，

语音 $s(n)$是滤波器的输入，而误差 $e(n)$是输出。事实上，系数$\{a_i\}$当然也需要从发端传到收端，但因为语音具有短时平稳性，即在短时间内(比如 10 ms)，系数$\{a_i\}$可以认为不发生变化，所以也不必太频繁地传输，因而采用预测技术后，总是可以大幅度地降低语音的带宽。这种通过线性预测方法压缩语音数据量的技术称为线性预测编码(Linear Prediction Coding，LPC)技术。

语音重建模型

如果知道激励信号 $x(n)$(先不考虑是如何得到的)和滤波器系数$\{a_i\}$，就可以利用激励生成重建语音了，和语音生成模型的公式完全相同

$$\hat{s}(n)=x(n)+\sum_{k=1}^{N}a_k\hat{s}(n-k)$$

但这个模型被称为语音重建模型，为了同生成模型区分开，$\hat{s}(n)$称为重建语音。如果 $x(n)$正好等于 $e(n)$，那么重建语音就会和原始语音 $s(n)$完全相同。

语音的非平稳性(虽然短时平稳)导致预测系数$\{a_i\}$是时变的，一般每 10～20 ms 就会发生一些变化以产生不同的音节。在这种情况下，滤波过程也要分段进行，即每次用不同的滤波器系数，但相邻两次滤波必须要保持滤波器的状态不发生变化。

谐振和共振峰频率

语音生成模型的每一对共轭极点都对应一个衰减的正弦信号的特征响应。例如一对共轭极点$|p_i|e^{\pm j\Omega}$在时域冲激响应中的贡献是$A|p_i|^n\cos(\Omega n+\varphi)$。其中极点幅度决定衰减速度，幅角决定振荡频率。

对语音合成，用数字的正弦信号表示抽样后的连续正弦信号。在这种情况下，模拟频率和数字频率的关系是 $\Omega=\omega T$，其中 T 表示抽样间隔，ω 表示模拟频率(弧度)，对应的 $f=\omega/2\pi$ 称为共振峰频率，它定义了声道的谐振频率。典型的男声($N=10$)可以用 5 个共振峰频率来描述。当模型参数变化时，共振峰频率也随着变化，从而产生不同的声调。

9.1.4 分析和合成语音

分析和合成系统如图 9.8 所示。

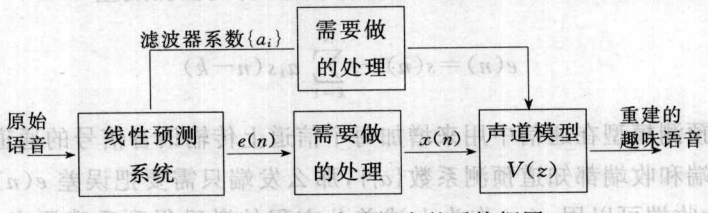


图 9.8 分析和合成语音的系统框图

首先要分析一段(一般是 10 ms)语音得到它的最佳$\{a_i\}$系数。给定这些系数后,就可以用适当的输入来合成语音。对于浊音信号,一种可取的激励模型就是以特定频率重复的单位样值序列,这个频率就是基音频率。对清音,最好选择随机噪声或者白噪声作为输入。但在不做清浊判决的情况下,全部采用周期激励的合成质量也可以接受(本章正是这么做的)。

分析过程如下:首先,抽样的语音信号被分成 10 ms 长的段;然后,对每段数据进行统计分析,计算相邻样点的相关性并最终得到最佳预测系数。合成过程就是利用这些预测系数,以及周期的单位样值序列作为输入,依次得到每段合成语音。

变速不变调

变速不变调技术广泛应用于消费类电子产品,如英语复读机等。所谓变速不变调,是指声音播放时,速度的改变不会导致音调的变化。一般来说,用随身听听录音,快进播放和慢速播放,其音调是不一致的,如快进播放,频率会变高,男声听起来会感觉是女声。(原因是什么?想一想$\Omega=\omega T$,如果Ω不变T却减小了,则ω会怎样?)为了实现变速不变调,首先需要将表示"调"的内容从语音中分离出来,由前述语音预测模型,表示"调"的有两个部分,一是共振峰频率,即预测模型的参数;二是基音周期,即激励信号的参数。接下来的工作就是在不改变这两种参数的前提下改变数据长度。即将 10 ms 的 80 个样点的激励变成 20 ms 160个样点(注意保持单位样值的周期不变),在这 20 ms 内保持预测模型系数不变,不就可以合成出 20 ms 的语音了吗?新语音的声调和原有语音是完全相同的,只不过时间变长了而已。

变调不变速

前面说过,最简单的男声变女声只要让随身听快进播放就可以了。但快进播放改变的不仅是声调,语速也会发生很大变化,快得让人听不清楚。为了解决这个问题,就需要用变调不变速的技术。同上述分析过程一样,还是需要在共振峰频率和基音周期上做改变。女声和男声的最大区别是频率高,一方面表现在基音频率高,另一方面共振峰对应的谐振频率也更高一些,所以我们可以考虑将激励信号的频率增加(注意不改变信号长度),同时将共振峰频率也相应增大一些(即极点的辐角绝对值增大,或者说上半平面的极点逆时针旋转,下半平面的极点顺时针旋转,但注意两者都要旋转同样角度而且不要转过负实轴)。这样得到的合成语音频率会更"女声"一些。

后两项技术是典型的语音信号数字处理技术。它们的基础是z变换和线性预测模型,用传统的模拟信号处理方法不可能实现,这正体现了数字信号处理的优点。最后还需指出,这两项技术并不矛盾,事实上,它们可以完美地结合在一起,你能做出一种速度和音调都发生变化的合成语音算法来吗?

第二节 语音合成综合实验

9.2.1 语音预测模型

(1) 给定

$$e(n)=s(n)-a_1 s(n-1)-a_2 s(n-2)$$

假设 $s(n)$是输入信号,$e(n)$是输出信号,上述滤波器的传递函数是什么? 如果 $a_1=1.3789$,$a_2=-0.9506$,上述合成模型的共振峰频率是多少? 用 zplane,freqz,impz 分别绘出零、极点分布图,频率响应和单位样值响应。用 filter 绘出单位样值响应,比较和 impz 的是否相同。

(2) 阅读 speechproc.m 程序,理解基本流程。程序中已经完成了语音分帧、加窗、线性预测和基音周期提取等功能。注意:不要求掌握线性预测和基音周期提取的算法原理。

```
function speechproc()
    % 定义常数
    FL = 80;                                  %帧长
    WL = 240;                                 %窗长
    P = 10;                                   %预测系数个数
    s = readspeech('voice.pcm',100000);       %载入语音 s
    L=length(s);                              %读入语音长度
    FN = floor(L/FL) - 2;                     %计算帧数
    %预测和重建滤波器
    exc = zeros(L,1);                         %激励信号(预测误差)
    zi_pre = zeros(P,1);                      %预测滤波器的状态
    s_rec = zeros(L,1);                       %重建语音
    zi_rec = zeros(P,1);
    %合成滤波器
    exc_syn = zeros(L,1);                     %合成的激励信号(脉冲串)
    s_syn = zeros(L,1);                       %合成语音
    %变调不变速滤波器
    exc_syn_t = zeros(L,1);                   %合成的激励信号(脉冲串)
    s_syn_t = zeros(L,1);                     %合成语音
    %变速不变调滤波器(假设速度减慢一倍)
```

```
exc_syn_v = zeros(2 * L,1);                %合成的激励信号(脉冲串)
s_syn_v = zeros(2 * L,1);                  %合成语音
hw = hamming(WL);                          %汉明窗
%依次处理每帧语音
for n = 3:FN
   %计算预测系数(不需要掌握)
   s_w = s(n * FL-WL + 1:n * FL). * hw;%汉明窗加权后的语音
   %A 是预测系数,E 会被用来计算合成激励的能量
   [A E] = lpc(s_w,P);                     %用线性预测法计算 P 个预测
                                           %系数
   if n == 27
   %(3)在此位置写程序,观察预测系统的零、极点分布图
   end
   s_f = s((n - 1) * FL + 1:n * FL);       %本帧语音,下面就要对它做
                                           %处理
   %(4)在此位置写程序,用 filter 函数 s_f 计算激励,注意保持滤
   %波器状态
   %exc((n - 1) * FL + 1:n * FL) = ...将你计算得到的激励写在这里
   %(5)在此位置写程序,用 filter 函数和 exc 重建语音,注意保持
   %滤波器状态
   %s_rec((n - 1) * FL + 1:n * FL) = ... 将你计算得到的重建语音
   %写在这里
   %注意下面只有在得到 exc 后才会计算正确
   s_Pitch = exc(n * FL - 222:n * FL);
   PT = findpitch(s_Pitch);                %计算基音周期 PT(不要求掌
                                           %握)
   G = sqrt(E * PT);                       %计算合成激励的能量 G(不要
                                           %求掌握)
   %9.2.2 节(4)在此位置写程序,生成合成激励,并用激励和 filter
   %函数产生合成语音
   %exc_syn((n - 1) * FL + 1:n * FL) = ...  将你计算得到的合成激
   %励写在这里
   %s_syn((n - 1) * FL + 1:n * FL) = ...    将你计算得到的合成语
   %音写在这里
   %9.2.3 节(1)不改变基音周期和预测系数,将合成激励的长度增
```

```
        %加一倍,再作为 filter 的输入得到新的合成语音,听一听是不是速
        %度变慢了,但音调没有变
        %exc_syn_v((n-1)*FL_v+1:n*FL_v)=...␣将你计算得到的
        %加长合成激励写在这里
        %s_syn_v((n-1)*FL_v+1:n*FL_v)=...␣将你计算得到的
        %加长合成语音写在这里
        %9.2.4 节(2)将基音周期减小一半,共振峰频率增加 150Hz,重新
        %合成听听感受
        %exc_syn_t((n-1)*FL+1:n*FL)=...␣将你计算得到的变
        %调合成激励写在这里
        %s_syn_t((n-1)*FL+1:n*FL)=...␣将你计算得到的变调
        %合成语音写在这里
    end
    %(6)在此位置写程序,听一听 s,exc 和 s_rec 有何区别,解释这种区别
    %后面听语音的题目也都可以在这里写,不再做特别注明
    %保存所有文件
    writespeech('exc.pcm',exc);
    writespeech('rec.pcm',s_rec);
    writespeech('exc_syn.pcm',exc_syn);
    writespeech('syn.pcm',s_syn);
    writespeech('exc_syn_t.pcm',exc_syn_t);
    writespeech('syn_t.pcm',s_syn_t);
    writespeech('exc_syn_v.pcm',exc_syn_v);
    writespeech('syn_v.pcm',s_syn_v);
end
%从 PCM 文件中读入语音
function s=readspeech(filename,L)
    fid=fopen(filename,'r');
    s=fread(fid,L,'int16');
    fclose(fid);
end
%写语音到 PCM 文件中
function writespeech(filename,s)
```

```
    fid = fopen(filename,'w');
    fwrite(fid,s,'int16');
    fclose(fid);
end
%计算一段语音的基音周期,不要求掌握
function PT = findpitch(s)
    [B,A] = butter(5,700/4000);
    s = filter(B,A,s);
    R = zeros(143,1);
    for k = 1:143
        R(k) = s(144:223)' * s(144 - k:223 - k);
    end
    [R1,T1] = max(R(80:143));
    T1 = T1 + 79;
    R1 = R1/(norm(s(144 - T1:223 - T1)) + 1);
    [R2,T2] = max(R(40:79));
    T2 = T2 + 39;
    R2 = R2/(norm(s(144 - T2:223 - T2)) + 1);
    [R3,T3] = max(R(20:39));
    T3 = T3 + 19;
    R3 = R3/(norm(s(144 - T3:223 - T3)) + 1);
    Top = T1;
    Rop = R1;
    if R2> = 0.85 * Rop
        Rop = R2;
        Top = T2;
    end if R3>0.85 * Rop
        Rop = R3;
        Top = T3;
    end
    PT = Top;
end
```

(3) 运行该程序到 27 帧时停住,用(1)中的方法观察零、极点分布图。

(4) 在循环中添加程序:对每帧语音信号 $s(n)$和预测模型系数$\{a_i\}$,用 filter 计算激励信号 $e(n)$。注意:在系数变化的情况下连续滤波,需维持滤波器的状

态不变，要利用 filter 的 zi 和 zf 参数。

(5) 完善 speechproc. m 程序，在循环中添加程序：用你计算得到的激励信号 $e(n)$和预测模型系数$\{a_i\}$，用 filter 计算重建语音 $\hat{s}(n)$。同样要注意维持滤波器的状态不变。

(6) 在循环结束后添加程序：用 sound 试听(5)中的 $e(n)$信号，比较和 $s(n)$以及 $\hat{s}(n)$信号有何区别。对比画出三个信号，选择一小段，看看有何区别。

9.2.2 语音合成模型

(1) 生成一个 8 kHz 抽样的持续 1 s 的数字信号，该信号是一个频率为 200 Hz 的单位样值“串”，即

$$x(n)=\sum_{i=0}^{NS-1}\delta(n-iN)$$

考虑该信号的 N 和 NS 分别为何值？用 sound 试听这个声音信号。再生成一个 300 Hz 的单位样值“串”并试听，有何区别？事实上，这个信号将是后面要用到的以基音为周期的人工激励信号 $e(n)$。

(2) 真实语音信号的基音周期总是随着时间变化的。我们首先将信号分成 10 ms 长的若干个段，假设每个段内基音周期固定不变，但段和段之间则不同，具体为

$$PT=80+5\ \mathrm{mod}(m,50)$$

其中 PT 表示基音周期，m 表示段序号。生成 1 s 的上述信号并试听。(提示：用循环逐段实现，控制每个段内每个脉冲和前一个脉冲的间隔为本段的 PT 值，注意每个段内的第一个脉冲要和上一(或多)个段的最后一个脉冲去比。)

(3) 用 filter 将(2)中的激励信号 $e(n)$输入到 9.2.1 节(1)的系统中计算输出 $s(n)$，试听和 $e(n)$有何区别。

(4) 重改 speechproc. m 程序。利用每一帧已经计算得到的基音周期和(2)的方法，生成合成激励信号 $Gx(n)$(G 是增益)，用 filter 函数将 $Gx(n)$送入合成滤波器得到合成语音 $\tilde{s}(n)$。试听和原始语音有何差别。

9.2.3 变速不变调

(1) 仿照 9.2.2 节(4)重改 speechproc. m 程序，只不过将 9.2.2 节(4)中合成激励的长度增加一倍，即原来 10 ms 的一帧变成了 20 ms 一帧，再用同样的方法合成出语音来，如果你用原始抽样速度进行播放，就会听到慢了一倍的语音，但是音调基本没有变化。

9.2.4　变调不变速

(1) 重新考察 9.2.1 节(1)中的系统，将其共振峰频率提高 150 Hz 后的 a_1 和 a_2 分别是多少？

(2) 仿照 9.2.2 节(4)重改 speechproc.m 程序，但要将基音周期减小一半，将所有的共振峰频率都增加 150 Hz，重新合成语音，听听是何感受。

MATLAB 知识点(16)——低级文件访问

为充分满足用户的各种要求，MATLAB 提供包括 fopen，fread，fwrite，fseek 和 fclose 等在内的一套直接访问二进制文件的函数，这些函数名和调用格式与 ANSIC 编程语言中的标准函数非常相似，请读者查阅帮助学习。

9.2.4 变调不变速

(1) 重新表示式 9.2.1 中(1)中的系数，将其共振峰频率提高 150 Hz 后的 α_i 和 σ_i 分别是多少？

(2) 仿照 9.2.2 节(4)重载 speechproc.m 程序，但要将基音周期减小一半，将所有的共振峰频率都增加 150 Hz，重新合成语音，听听是何感受。

MATLAB 知识点(16)——任意文件访问

为充分满足用户的各种要求，MATLAB 提供了如 fopen, fread, fwrite, fseek 和 fclose 等内的一套直接访问二进制文件的函数，这些函数名和调用格式与 ANSIC 编程语言中的标准函数非常相似，请读者查阅帮助学习。

第四篇

MATLAB 编程辅导二

第四篇

MATLAB 编程指导二

第十章　高级编程知识

第一篇讲授的 MATLAB 基本编程方法，已经通过第二、三篇中的例题和大作业进行练习而得到巩固。本章将介绍一些高级编程知识，包括函数、变量的分类和作用域、使用函数句柄简化程序设计等内容。

第一节　函数和变量

一个函数 M 文件中可以定义多个函数，其中第一个函数称为主函数，随后的函数称为子函数，主函数和子函数内部都可以定义嵌套式函数，此外还有局部函数和匿名函数，下面依次解释这些函数的意义和使用方法。理解匿名函数需要知道函数句柄，因而将在下一节讲解匿名函数。

主函数

作为 M 文件中的第一个函数，主函数是该 M 文件的代表，调用该文件名时执行的即是主函数。一般而言，主函数名和文件名相同，这就明确表征了主函数的意义；如果主函数名和文件名不同，无论其后有无和文件名相同的子函数，调用该文件名时调用的仍然是主函数而不是任何一个子函数。MATLAB 是解释性的程序语言决定了这种调用方法。

子函数

MATLAB 支持在 M 文件的主函数之后放置任意数量的子函数。子函数的定义方式和主函数完全相同，它用以实现主函数中的特定功能，改善 M 文件的结构和层次，实现代码复用，又避免建立额外的 M 文件带来维护的麻烦。子函数和主函数的根本区别是：只能为主函数或者同一 M 文件中的其他子函数调用，而不可以在该文件外的 M 文件或者命令窗口中调用。

嵌套式函数

第一章中介绍过函数以 function 开始，顺序执行到文件最后一行、下一个函

数开始、或者 end 时结束。通过 end 标识，MATLAB 支持嵌套式函数，即在 A 函数内部嵌套定义 B 函数，B 函数只能在 A 函数内被调用，而在 A 函数外其他任何地方都是不可见的。嵌套函数的引入是为了缩小子函数的存在空间，避免子函数在 M 文件内的其他地方被误调用。MATLAB 为了避免解析子函数和嵌入式函数时可能导致的混乱，特别要求：在同一个文件中，要么所有函数都以 end 标识结束，要么所有函数都不要 end，不能两者混合使用。

局部函数

引入子函数是为了代码复用和程序结构清楚，同时避免被其他函数文件调用。如果同一文件内子函数过多，或者某子函数代码很长，可以将这些子函数改成局部函数。局部函数放置在 private 目录下，只能被和该 private 目录同级的函数调用。当 MATLAB 解释程序根据名称寻找函数时，首先检查是否为同文件内部的子函数，再在 private 目录下检查是否为局部函数，如果都找不到，最后才在搜索路径列表中寻找。

以上介绍的函数类型都是硬盘中存在的 M 文件。除了带有源码的函数外，为了提高常用函数的运行速度，MATLAB 还定义了一种没有源码的内部(built-in)函数。例如在命令窗口中输入

```
type filter
```

MATLAB 将回显

```
filter is a built - in function.
```

结合函数，MATLAB 的变量可分为输入变量、输出变量和函数内部使用的变量。本书前面介绍的基本都是函数内部使用的变量。输入变量是函数的入口数据，是作用在函数内部的局部变量，函数内对输入变量的修改不会在函数返回后带来影响。函数对输入变量的操作只有依靠输出变量才能得以体现。

输入输出变量

函数可以没有输入输出变量，也可以有任意多的输入输出变量。MATLAB 在调用函数时只对输入参数和输出参数的数量进行检查，它允许两者分别小于函数定义中输入变量和输出变量的数量。在函数内部，可以通过 nargin 和 nargout 两个函数分别获取输入参数和输出参数的真实数量。进入函数内部后，输入参数并没有被复制到函数的工作空间，但是它们仍然可以被访问，但修改结果不会反映到调用函数的工作空间。

全局变量

函数内部定义的变量都是局部变量，它们不能在子函数中或者该函数返回后使用。MATLAB 提供了 global 函数声明全局变量，即在使用该全局变量的任何函数内部进行声明。若要在 base 工作空间中使用，也需要在命令窗口中声明。

永久变量

全局变量延长了变量的生存期和使用空间，但不可避免地增大了变量被误修改的可能性。为了保护全局变量只在特定函数中使用，可以用 persistent 声明变量为永久变量，即可以在声明的函数工作空间中使用，函数返回并再次进入后该变量值仍然保持，但也只可以在这些函数空间中使用。

第二节　函数句柄

和图形对象的句柄类似，函数句柄提供了访问函数的另一种途径：可以将函数定义为数组成员，根据数组下标访问不同的函数，从而减小程序设计中的冗余。用 hf＝@functionname 定义 hf 为函数 functionname 的函数句柄，通过句柄执行该函数时只要调用 hf(param1,param2,...)即可。func2str 和 str2func 可以在函数句柄和字符串之间相互转换。

为了定义短函数，同时避免新建 M 文件或者函数声明的麻烦，MATLAB 引入了匿名函数的概念。匿名函数通过函数句柄调用，其定义格式为 hf＝@(param1,param2,...)expression，其中 param1 等是函数的输入参数，expression 为实现该函数功能的 MATLAB 表达式，该匿名函数的句柄将返回 hf 中。因为唯独没有声明函数名，所以称为匿名函数。

下面举例演示函数句柄和匿名函数的定义和调用方法。

例 10.1　在四个子图中分别绘制 $\sin t$、$\cos t$、e^t 和 t^2-4t+1 四个信号在 $t\in[0,2\pi]$ 区间的波形。

解　§§§MATLAB 文件 ex_10_1.m

```
t = [0:0.1:2 * pi]';                    %定义抽样时间
fh = {@sin,@cos};                       %先定义两个函数的句柄为单元数组
fh{3} = str2func('exp');                %第三个句柄用字符串定义
fh{4} = @(t)t.^2 - 4 * t + 1;           %第四个句柄用匿名函数定义
figure;
for n = 1:4                             %依次绘制四个信号波形
    subplot(2,2,n);                     %选择子图
    plot(t,fh{n}(t));                   %绘图 fh{n}即为第 n 个函数
    xlabel('t');                        %设置 X 坐标文字
    title([func2str(fh{n}),'(t)']);     %设置标题文字为函数名
    set(gca,'XLim',[0,2 * pi]);         %设置 X 坐标范围
end
```

程序运行结果如图 10.1 所示。通过使用函数句柄，本来需要重复四次的命

令只需要在循环中实现即可。

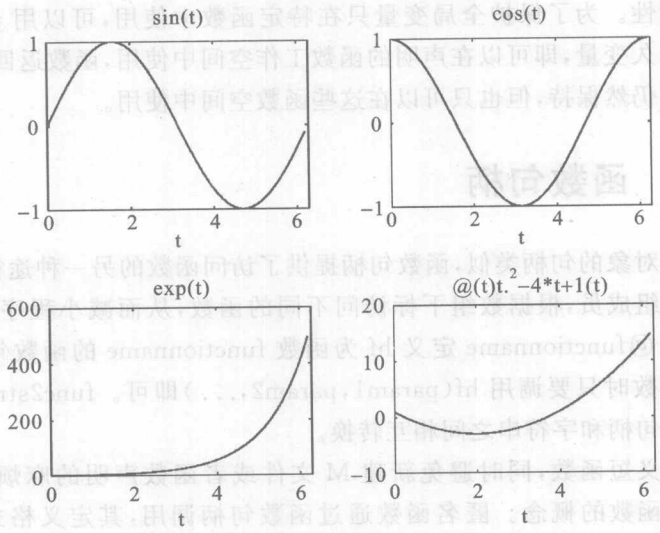


图 10.1 例 10.1 运行结果

第十一章　Simulink 仿真

Simulink 是 MATLAB 的一个软件包，用来实现建模、仿真和动态系统分析。它的特点是使用简便但功能强大，通过图形用户界面建模，可以仿真连续时间系统、离散时间系统或者两者的混合系统，因而是信号与系统实验的重要工具。本章将通过一道例题详细讲解 Simulink的各种使用方法。

第一节　启动 Simulink

启动 Simulink，在 MATLAB 命令窗口中输入“simulink”，或者在 MATLAB工具条中点击 Simulink 按钮，都可以启动 Simulink Library Browser 窗口，如图 11.1 所示。

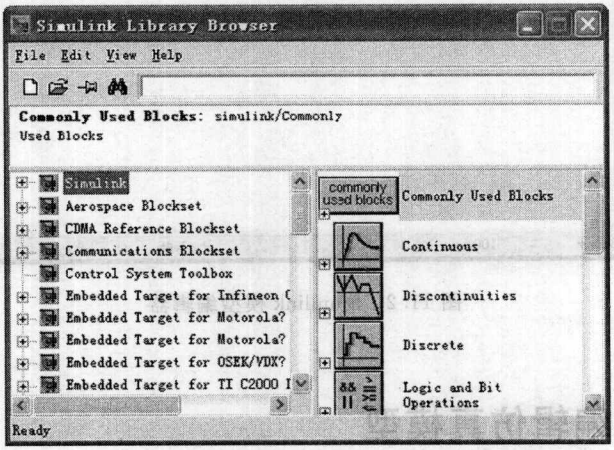


图 11.1　Simulink 类库浏览器

Simulink Library Browser窗口主要有三部分:左下方树状视图显示的是本机所安装的Simulink模块库;右下方列表视图显示当前选中的库里的所有模块,你可以用鼠标左键拖拽或者复制它们到你自己的模块编辑器中;上方只读编辑窗口显示当前被选中的模块的说明。最上方狭长的编辑窗口用于搜索模块,你只要在这里输入模块名,再按回车键或者前面的搜索按钮,都可以找到该模块并且在下方的两个视图中显示。

第二节 建立、打开和保存仿真模型

如果说MATLAB工作的对象是M文件,那么Simulink工作的对象就是仿真模型(Model)。在Simulink Library Browser中选择File→New→Model,或者直接点击工具条上的"新建"按钮,都会建立一个未命名的模型,并在模型编辑器窗口中打开,如图11.2所示。此时即可选择File→Save并为其指定名称保存,注意模型的文件名后缀是.mdl。下次需要再次打开该模型时,只要在Simulink Library Browser窗口或者模型编辑器窗口中,选择File→Open浏览到该文件打开即可。

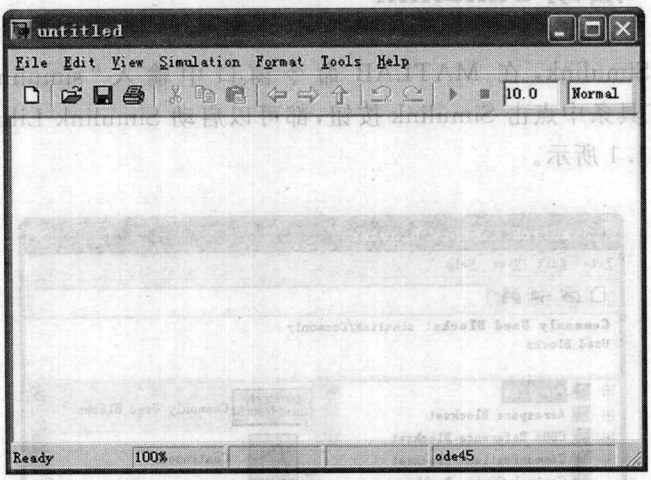


图11.2 Simulink模型编辑器

第三节 编辑仿真模型

一个新建的空白模型中需要插入各个模块才能完成需要的任务,现以一个

电路系统的仿真例题说明如何编辑模型。

例 11.1(主教材例 5-2 修改) 图 11.3(a)所示 RC 低通网络,在输入端 1-1'加入矩形脉冲 $v_1(t)$如图 11.3(b)所示,利用傅里叶分析方法求 2-2'端电压 $v_2(t)$。图中 $E=1$ V,$\tau=0.5$ s。

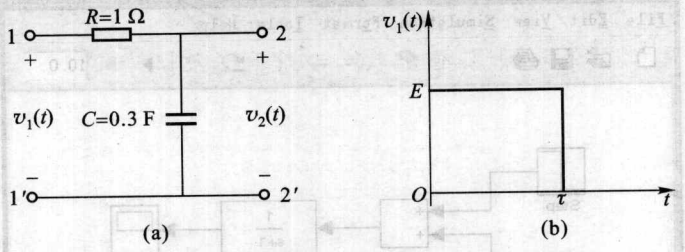


图 11.3 例 11.1 电路图和激励波形

解 仿真模型一般包括三个组成部分——信号发生器、处理器和信号接收器。信号发生器一般在 Simulink\Sources 库中,但经查找未找到所需的矩形脉冲信号发生器,却发现了阶跃信号发生器 Step,所以将输入信号变形为

$$v_1(t)=u(t)-u(t-0.5)$$

即除了两个阶跃信号发生器之外,还需要一个加法器 Add,在 Simulink\Math Operations 库中。本例的处理器比较简单,是一个连续时间传递函数模块,可以在 Simulink\Continuous 库中找到 Transfer Fcn 完成该功能。信号接收器一般都在 Simulink\Sinks 库中,浏览后发现用示波器 Scope 即可。

现在开始搭建例题仿真模型。首先新建一个模型,保存为SSExample. mdl。然后依次从 Simulink Library Browser 中找到所需的模块,用鼠标左键拖拽到模型编辑器中(或者在该模块上点击鼠标右键,选择 Add to SSExample),根据输入在左侧、输出在右侧的一般顺序放置好这些模块,并在相互之间保持一定的距离。接下来开始在模块的输入和输出端口之间直接建立连接,有两种连接方法:一是用鼠标左键点中一个模块输入/输出端后拖拽到另一个模块的输出/输入端,二是先用鼠标点击(带有输出端的)源模块,选中它,然后按住 Ctrl 键不放,再用鼠标点击(带有输入端的)目标模块,这样便会有线将两者的输出和输入端连接起来。第二种方法对于有多个输入/输出端的模块也是有效的,Simulink 会顺序连接空闲的第一个输入/输出端。如果不小心连接错了,可以先选中错误的连接线,然后按 Delete 键将其删除。连接有问题的线会以红色虚线显示,建议先删除这种线,然后再重新按照正确连接方法绘制。每个模块图标的大小和名称都可以修改,只要用鼠标拖拽边框或者用左键双击该模块下方的文字即可。

如果希望在模型中添加一些文字说明或标记，也可以在任何空白的地方双击鼠标左键，然后输入要添加的文字。最后搭建好的模型如图11.4所示，对每个模块都保留了它们默认的名字。

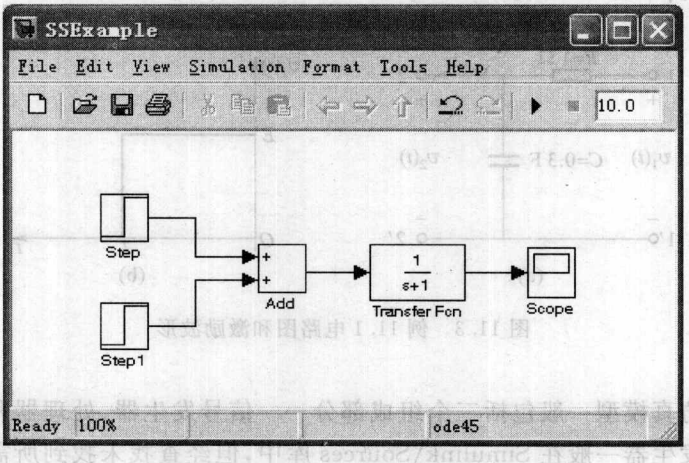


图11.4 例11.1仿真模型(未设置参数)

接下来需要配置各个模块的参数。首先是输入信号，双击Step图标(注意不是下边的Step文字)，会弹出图11.5所示的Source Block Parameters:Block对话框，对话框首先解释了该模块输出一个阶跃信号，然后列出了Step模块的各个参数供设置:Step time表示发生跳变的时间，Initial value和Final value分别表示跳变前后的值，Sample time表示生成信号的抽样时间，默认为0。根据题意，将这两个Step模块的参数分别设置成:0时刻从0跳变到1和0.5时刻从0跳变到−1，这样两个阶跃信号之和即可构成需要的方波输入信号。若要了解Step模块的具体说明和参数意义，可以在Simulink Library Browser中选中Step模块，点击鼠标右键后选择Help for the Step block，或者在模型编辑器中用鼠标右键点击某个Step模块后选择Help。Add模块选用默认配置即可，不再赘述。

接下来看连续时间传输函数模块Transfer Fcn，双击该模块将弹出如图11.6所示的属性窗口，其中前两个编辑框分别用来输入传递函数分子和分母多项式的系数，最后一个Absolute tolerance默认为auto，其具体意义请参考帮助文件。根据题意，系统传递函数应为 $H(s)=\dfrac{1}{0.3s+1}$，所以在两个编辑框中分别输入“[1]”和“[0.3,1]”。

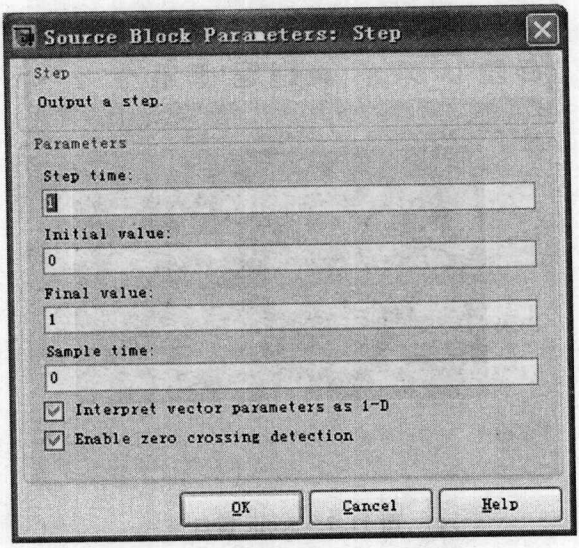


图 11.5 Step 模块参数设置窗口

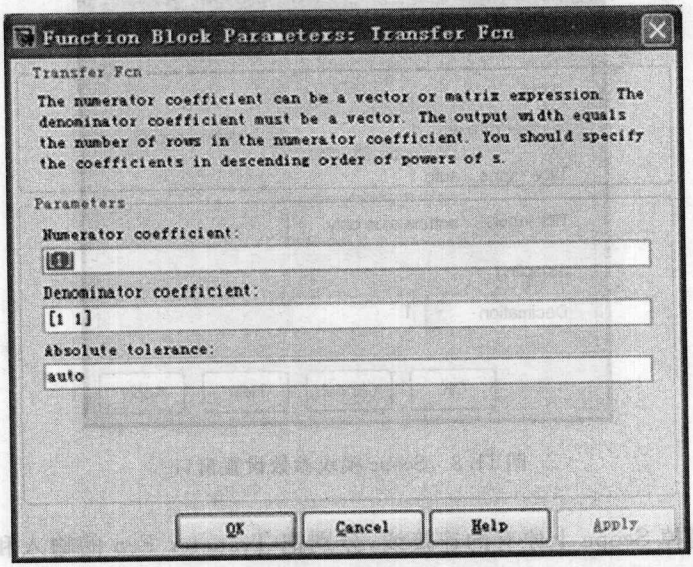


图 11.6 Transfer Fcn 模块参数设置窗口

最后看示波器模块 Scope，双击后显示图 11.7，再点击其工具条上左数第二个 Parameters 按钮，弹出如图 11.8 所示的属性窗口，因为往往需要在查看输出信号的同时对比输入信号，所以把属性 Number of axes 改为 2，其他参数不变，点击确定，即可看到刚才只有一个坐标系的示波器屏幕变成了上、下两个坐标

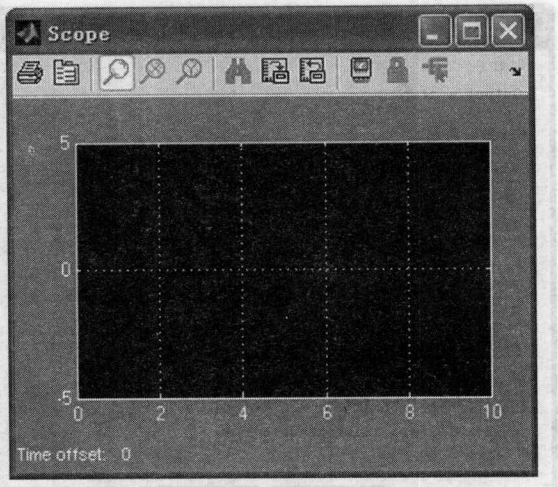


图 11.7 Scope窗口

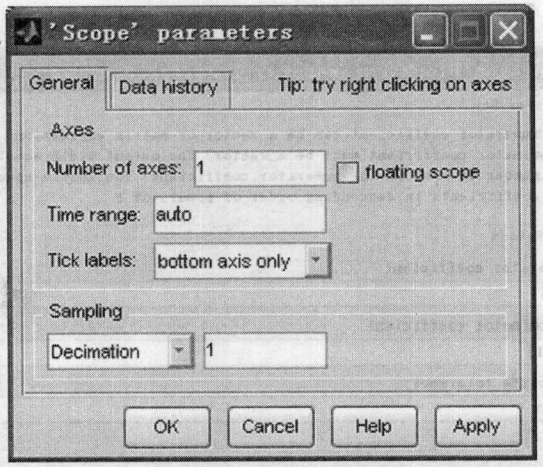


图 11.8 Scope模块参数设置窗口

系。现在删掉 Scope 上原有的连接线，分别把 Transfer Fcn 的输入和输出连接到 Scope 的两个输入上，即可看到编辑完成的模块最终如图 11.9 所示。

对比图 11.4 和图 11.9，可见 Step1 的模块图标内波形由上跳改为下跳，Transfer Fcn 的模块图标内公式显示为 $H(s)=\dfrac{1}{0.3s+1}$，从而可以深刻感受到 Simulink 界面的友善性。

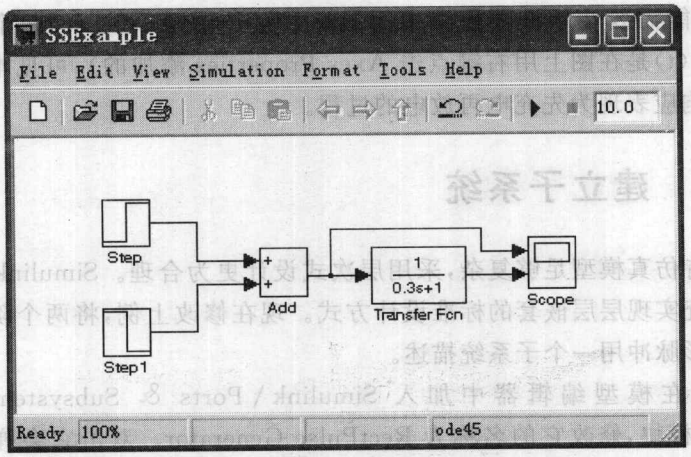


图 11.9　例 11.1 仿真模型(参数设置完毕)

第四节　运行仿真模型

编辑好仿真模型后即可准备运行,Simulink 认为从零时刻仿真开始,结束的时间由用户在模型编辑器的工具条上的编辑框中输入,单位是秒,把默认值 10.0 改为 2.0。这时可以选择 Simulation→Start 或者用鼠标点击工具条上的“开始”按钮启动仿真。仿真过程中该按钮变灰,旁边的“结束”按钮变亮从而可以随时停止仿真。仿真结束后,“开始”按钮恢复变亮。

仿真结束后,点击 Scope 再次打开示波器窗口,会看见如图 11.10 所示的仿

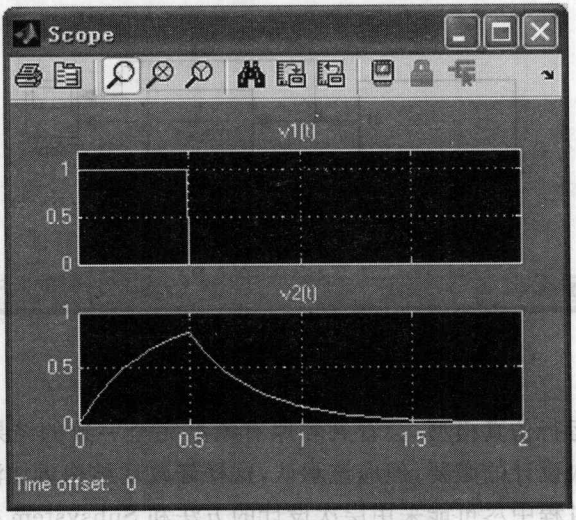


图 11.10　例 11.1 仿真结果

真结果(可能需要在示波器窗口中用鼠标右键选择 Autoscale,两个图的标题 v1(t)和 v2(t)是在图上用右键点击 Axes Properties 添加的),可见本系统对矩形脉冲的响应表现为先充电再放电的过程。

第五节 建立子系统

如果待仿真模型足够复杂,采用层次式设计更为合理。Simulink 支持用自定义子系统实现层层嵌套的标准设计方式。现在修改上例,将两个阶跃信号相加生成矩形脉冲用一个子系统描述。

首先在模型编辑器中加入 Simulink\Ports & Subsystems 库中的 Subsystem模块,修改它的名称为 RectPulse Generator。双击它会弹出一个新的模型编辑器窗口,标题为 SSExample/RectPulse Generator,回到原来的窗口,用鼠标左键拉框选中两个 Step 模块和一个 Add 模块,按 Ctrl+x 键剪切,再切换回 RectPulse Generator 的窗口,按 Ctrl+v 键,将上述模块粘贴过来,然后删掉RectPulse Generator中原有的 In1 端口,把 Out1 改名为 v1,并把 Add 的输出和 v1 端口连接起来,这样两个模型编辑窗口分别如图 11.11 和图 11.12 所示。

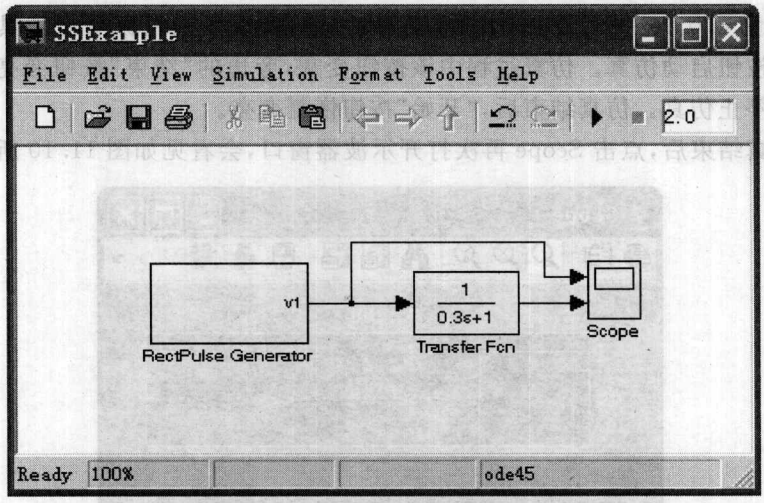


图 11.11 例 11.1 仿真模型(带子系统)

现在再次运行仿真模型,会看到和原有模型完全一致的结果。虽然本例完全没有采用层次设计的必要,但应该承认,这样修改后结构更加清晰了。因而建议读者在设计过程中尽可能采用层次设计的方法和 Subsystem 模块。

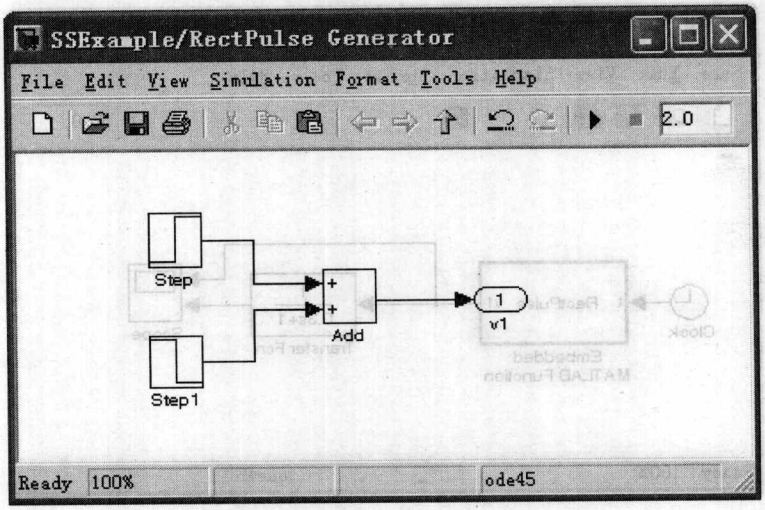


图 11.12 例 11.1 仿真模型(子系统)

第六节 利用 MATLAB 函数和程序

虽然 Simulink 功能强大且资源丰富,但仍然可能有尚未支持的模块或者无法完成的任务,这时候就需要手工编写 MATLAB 程序嵌入到 Simulink 模型中;或者有时候已经在 MATLAB 中完成了很多任务,为了以最小工作量移植到 Simulink 中,也希望尽可能多的保留原来的 MATLAB 程序。本例中的各个模块虽然都可以由 Simulink 胜任,但仍然可以把其中的个别模块用 MATLAB 编程实现。例子中最复杂的部分是矩形脉冲生成单元,下面将用MATLAB 文件实现它。

首先删掉刚才自定义的 RectPulse Generator 模块,再把 Simulink\User-Defined Functions 库中的 Embedded MATLAB Function 和 Simulink\Sources 中的 Clock 模块拖拽到编辑窗口,把 Embedded MATLAB Function 的输入连接 Clock 的输出,它的输出连到 Transfer Fcn 的输入上。

现在双击 Embedded MATLAB Function 模块,会弹出一个 Enbedded MATLAB Editor 的窗口,删除其中内容并输入如下命令

```
function v1 = RectPulse(t)
v1 = double(t>0&t<0.5);
```

现在关闭该编辑窗口。最终模型编辑器如图 11.13 所示。

对以上模型进行仿真后,MATLAB 命令窗口中将出现

```
Embedded MATLAB parsing for model "SSExample"... Done
```

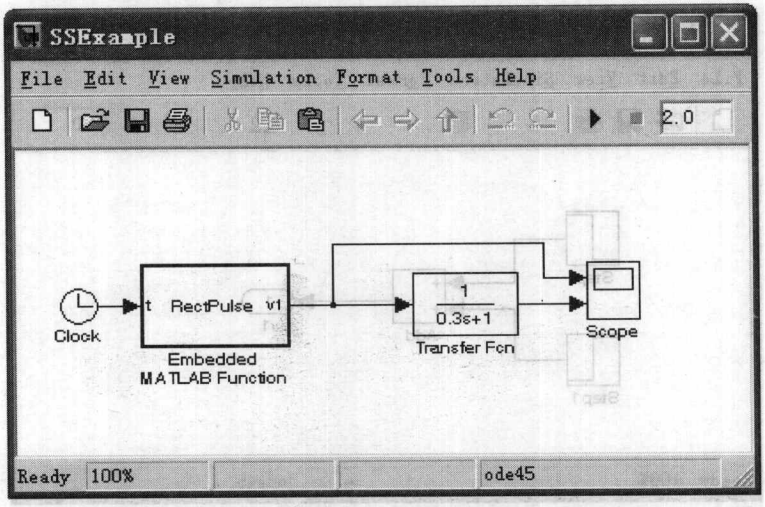


图 11.13 例 11.1 仿真模型(自定义函数)

```
Embedded MATLAB code generation for model "SSExample". . . . Done
Embedded MATLAB compilation for model "SSExample"... 已复制 1 个文件。
Done
```

可看到 MATLAB 对刚才编辑的程序进行解析和编译并最终植入 Simulink 的过程。最后 Scope 的结果如图 11.14 所示。由于 Clock 模块仿真时间间隔较

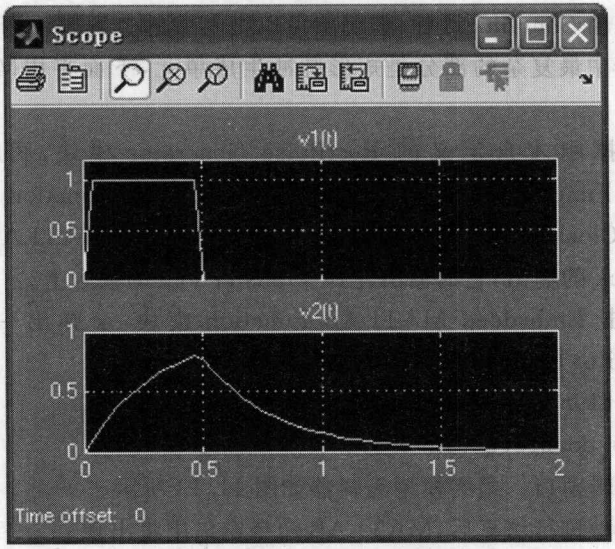


图 11.14 例 11.1 仿真结果(自定义函数)

大，所以矩形脉冲和充、放电波形都有些不连续。请注意这里只支持嵌入式 MATLAB 语言，即标准 MATLAB 命令的一个子集，相当多的外部命令，比如绘图操作等都不支持。

像本例这样，只是调用 MATLAB 的函数或者一句 MATLAB 命令，也可以使用 Simulink\User-Defined Functions 库中的 Fcn 模块，其使用方法更为便捷，读者可自行练习。

第七节 访问工作空间中的变量和硬盘上的数据文件

为支持用 Simulink 访问 MATLAB 工作空间中的变量，Sources 库和 Sinks 库分别提供了 From Workspace 和 To Workspace 两个模块。同理，这两个库还提供了 From File 和 To File 两个模块用以和硬盘上的扩展名为“.mat”的标准数据文件交互。

再次修改前述例题以演示上述功能。首先在 MATLAB 命令窗口中定义如下变量

```
t = [0:0.01:2]';
x = (t<0.5);
v1 = [t,x];
```

这里定义了一个变量 v1，它的第一列是抽样时间 t，第二列是矩形脉冲数据。然后编辑模型，添加 Simulink\Sources 库中的 From Workspace 和 Simulink\Sinks 中的 To File 两个模块到编辑器中，删掉其他无用模块并连接好输入输出端口。双击 From Workspace 模块显示其参数，阅读其中的说明文字可知对一维输入信号要求第一列为抽样时间，第二列为数据，上面定义的格式满足要求，将参数中的 Data 改为 v1。再双击 To File 模块，阅读说明发现它对数据是以行矢量格式保存，第一行为抽样时间，将参数 Filename 改为 v2.mat，Variable name 改为 v2，其他参数保持不变。最后形成如图 11.15 所示的仿真模型。

现在运行仿真模型，结束后在命令窗口中输入

```
clear all
load v2.mat
plot(v2(1,:),v2(2,:));
```

即得到图 11.16 所示波形，可见与图 11.10 完全相同。

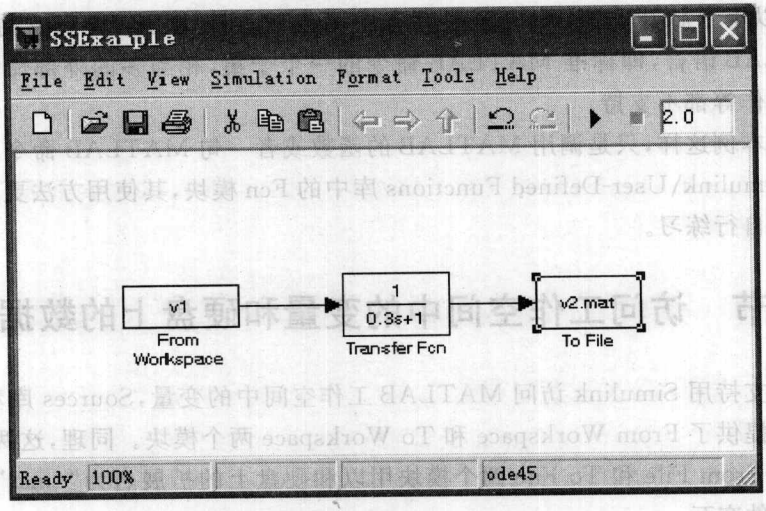


图 11.15 例 11.1 仿真模型(访问 MATLAB 资源)

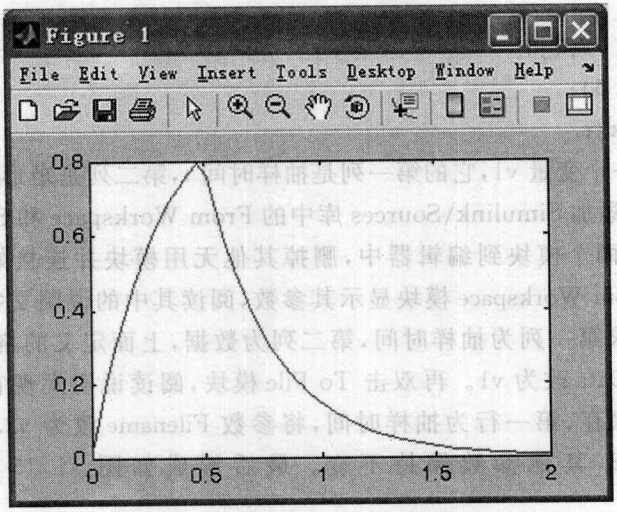


图 11.16 例 11.1 仿真结果(从 MATLAB 文件中读入)

第八节 Simulink 支持的库和模块

以上演示了 Simulink 的基本使用方法,更多的内容请参考联机帮助文档。为了用 Simulink 搭建模型仿真解决问题,必须首先知道 Simulink 有哪些资源,这就需要读者认真地浏览 Simulink Library Browser 以充分了解更多支持库和模块。

除了常用的 Simulink 基本库，还有包括 Communications Blockset 在内的数十个 Blockset 或者 Toolbox。本书后面主要用到 Communications Blockset 和 CDMA Reference Blockset，以及 Signal Processing Blockset 三个库中的若干模块，请读者在自学时间多多浏览。

最后需要指出，快速便捷地使用 Simulink 完成仿真工作是建立在扎实的理论基础之上的。不要误以为简单地拖拽模块、修改参数就可以解决问题。加强“信号与系统”理论学习，在坚实的理论基础上培养动手能力，是成为未来优秀信息产业人才的必由之路。

第十二章　高级绘图技术

本章讲解 MATLAB 绘制特殊图形和进行高级图形控制的方法。虽然这部分内容在“信号与系统”中使用较少，但下面将通过“信号与系统”的典型例题说明一些复杂抽象的问题可以用三维图形等高级绘图技术方便地阐述。

第一节　三维绘图和特殊图形

学过三大变换(傅里叶变换、拉氏变换、z 变换)后，可知虚轴上的拉氏变换即是傅里叶变换，而单位圆上的 z 变换就是离散时间傅里叶变换(DTFT)。下面通过两道例题分别展示拉氏变换和傅里叶变换的关系，以及 z 变换和 DTFT 的关系。

例 12.1　对矩形脉冲信号

$$x(t)=\begin{cases}1 & 0<t<10\\ 0 & \text{其他}\end{cases}$$

做拉氏变换和傅里叶变换，绘图说明两者之间的关系。

解　§§§MATLAB 文件 ex_12_1.m

```
syms t x
x = heaviside(t)-heaviside(t - 10);          %用定义矩形脉冲
                                             %的符号函数

Fx = fourier(x);                             %做傅里叶变换
omg = [ - 3 * pi:0.02:3 * pi];               %定义频率抽样点
Fx1 = subs(Fx,'w',omg);                      %计算抽样点上的傅
                                             %里叶变换值

Lx = laplace(x);                             %做拉氏变换
sgm = [0:0.05:1];                            %定义 σ 抽样点
```

```
s1 = log(kron(exp(sgm),exp(j * omg')));        %定义 s 平面上的
                                               %抽样点s=σ+jω
Lx1 = subs(Lx,'s',s1);                         %计算抽样点上的拉
                                               %氏变换值
figure,box on,hold on,grid on;                 %生成新图框
surf(sgm,omg,real(Lx1),'linestyle','none');    %绘制拉氏变换的
                                               %三维曲面
plot3(0 * omg,omg,real(Fx1),'k','LineWidth',2); %绘制傅里叶变换的
                                               %三维曲线
xlabel('\sigma','FontSize',14);                %x 坐标标注 σ
ylabel('\omega','FontSize',14);                %y 坐标标注 ω
zlabel('L(x) & F(x)','FontSize',14);           %z 坐标标注两个变换
set(gca,'FontSize',14);                        %修改字体和坐标刻度
set(gca,'XTick',[0:0.5:1],'YTick',[-2 * pi:pi:2 * pi]);
set(gca,'YTickLabel',['-2pi';'-pi';'0';'pi';'2pi']);
```

程序运行结果如图 12.1 所示，可见虚轴上的拉氏变换就是傅里叶变换以 2π 为周期重复后叠加的结果。

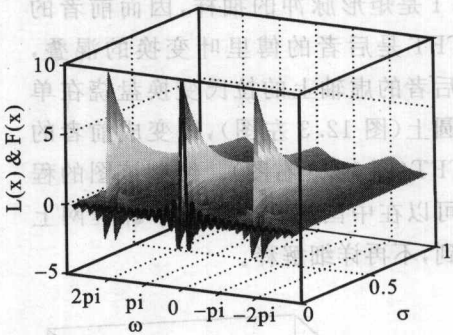


图 12.1　例 12.1 运行结果

例 12.2　对矩形序列

$$x(n)=\sum_{i=0}^{9}\delta(n-i)$$

做 z 变换和离散时间傅里叶变换，绘图说明两者之间的关系。

解　§§§MATLAB 文件 ex_12_2.m

```
syms n x w
x = heaviside(n)-heaviside(n-10);              %定义矩形序列的符号函数
Zx = ztrans(x);                                %做 z 变换
DTFTx = subs(Zx,'z',exp(j * w));               %以 e^jω替换 z 即 DTFT
r = [0.1:0.01:1.5];                            %定义极坐标的模
w1 = [0:0.01:2 * pi + 0.1];                    %定义极坐标的辐角
z1 = kron(r,exp(j * w1)');                     %生成 z 平面上的抽样点
Zx1 = subs(Zx,'z',z1);                         %计算抽样点的 z 变换值
DTFTx1 = subs(DTFTx,'w',w1);                   %计算单位圆上抽样点的DTFT
```

```
figure,box on,hold on,grid on;                    %生成新图框
Zx2 = max(min(real(Zx1),15), - 5);                %截取实部[-5,15]区间的值
DTFTx2 = max(min(real(DTFTx1),15), - 5);
surf(real(z1),imag(z1),Zx2,'linestyle','none');   %绘制 z 变换曲面
plots3(cos(w1),sin(w1),DTFTx2,'k','LineWidth',2);%绘制 DTFT 曲线
xlabel('z_{real}','FontSize',14);                 %标示等操作,无需
                                                  %赘述
ylabel('z_{imag}','FontSize',14);
zlabel('z(x) & DTFT(x)','FontSize',14);
set(gca,'FontSize',14);
set(gca,'XLim',[ - 1.5,1.5],'YLim',[ - 1.5,1.5]);
```

程序运行结果如图 12.2 所示,可见单位圆上的 z 变换就是 DTFT。

对比以上两个例题,还可以研究傅里叶变换以及拉氏变换和 DTFT 的关系。例 12.2 中的矩形序列是例 12.1 是矩形脉冲的抽样,因而前者的 DTFT 是后者的傅里叶变换的混叠,而后者的虚轴上的拉氏变换盘绕在单位圆上(图 12.3 左图),即变成前者的 DTFT(图 12.3 右图)。绘制该图的程序可以在中国高校电工电子课程网上找到,不再详细解释。

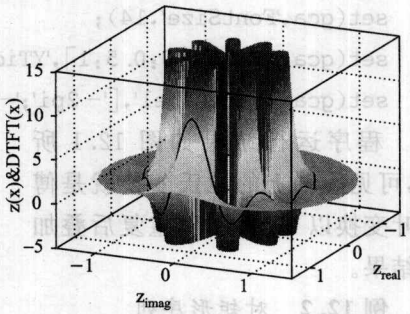


图 12.2 例 12.2 运行结果

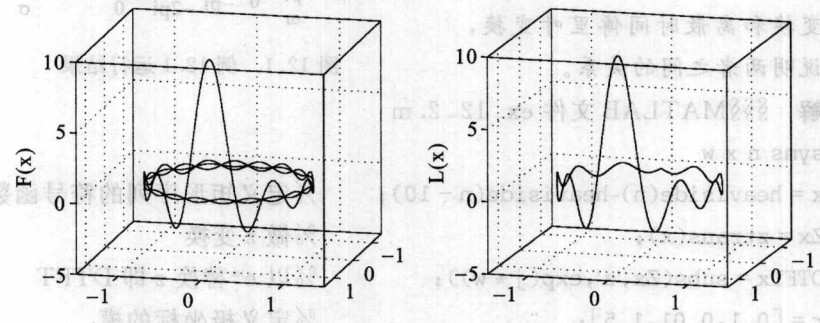


图 12.3 傅里叶变换、拉氏变换和离散时间傅里叶变换之关系

上面例题演示了 surf(曲面)和 plot3(三维曲线)两个函数的使用方法和绘图效果,除此之外,MATLAB 还提供了 contour(等高线)、waterfall(瀑布线)、

contour3(三维等高线)和 mesh(网格面)等函数分别用于绘制三维图形,所有上述函数的演示效果如图 12.4 所示,请参考中国高校电工电子课程网上的程序自学。

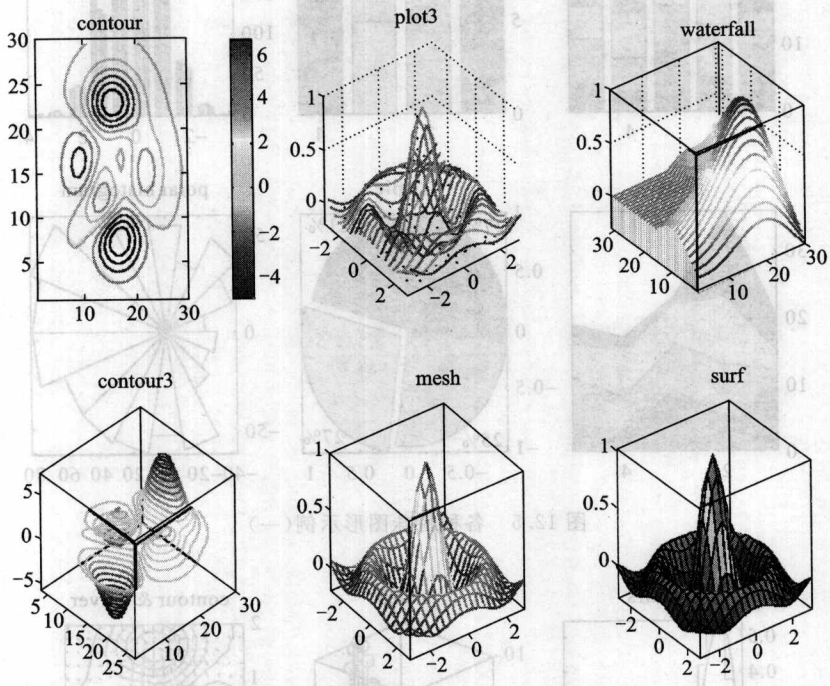


图 12.4　各种 3D 图形示例

除了三维绘图函数外,MATLAB 还提供了各种特殊绘图函数,包括 bar(柱状图)、bar3h(三维柱状图)、hist(直方图)、area(面积图)、pie3(三维饼图)、rose(极坐标直方图)、stairs(零阶抽样保持图)、stem3(三维序列图)、quiver(场强图)、polar(极坐标图)、compass(原点指向图)和 feather(线性指向图)函数等,分别用于绘制各种特殊图形。所有上述函数的演示效果如图 12.5 和图 12.6 所示,读者不需要记住这些命令,但要知道 MATLAB 都支持哪些特殊图形,以便在需要时使用。

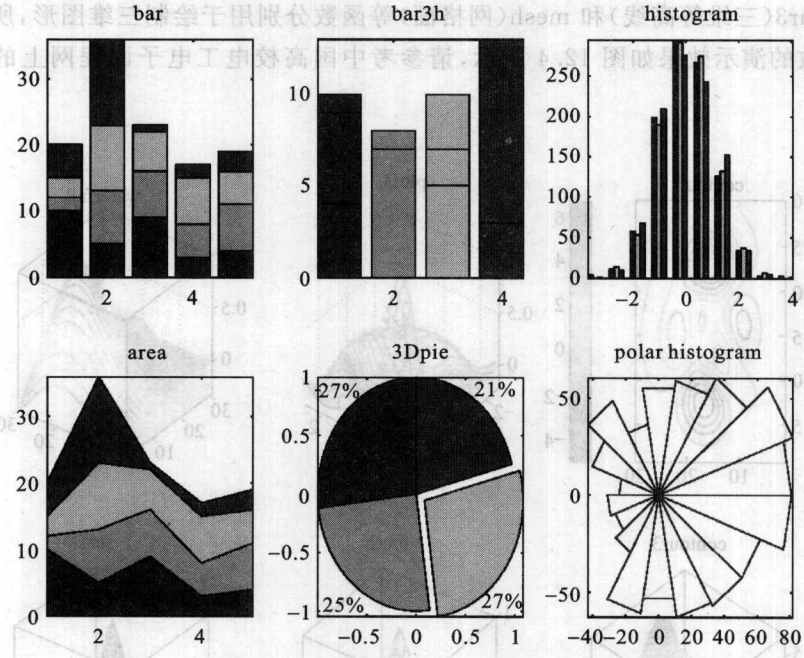


图 12.5 各种特殊图形示例(一)

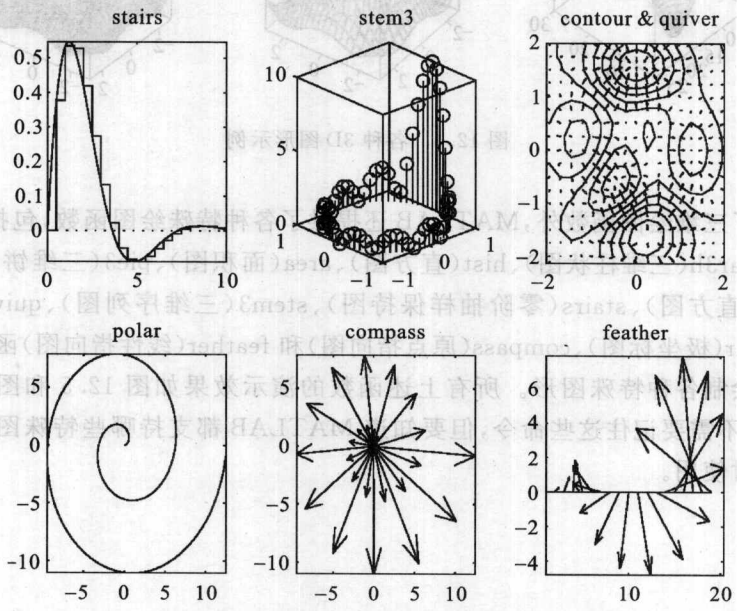


图 12.6 各种特殊图形示例(二)

第二节　图形高级控制

MATLAB 的绘图功能在颜色和光照控制、视点控制和图形旋转等方面也提供了强大的支持。这部分内容在电子工程领域使用较少，但在此还是编制了一条示例程序供读者学习，程序的名字叫“太阳照耀在 z 平面上”。程序为动画演示，请读者自己运行并根据帮助学习程序。

§§§MATLAB 文件 ex_12_3.m

```
syms z1 w n
z1 = ztrans(sin(n),w);
X = [ - 3:0.05:3];
Y = [ - 3:0.05:3];
Z = subs(z1,'w',log(kron(exp(x),exp(j * Y'))));

figure;
hold on;
grid off;
colormap(hot);
surf(X,Y,real(Z),'linestyle','none');
W = 4;
set(gca,'XLim',[ - W W],'YLim',[ - W W],'ZLim',[ - W W]);
set(gca,'XTick',[],'YTick',[],'ZTick',[]);

R = 3;
ele = 60;
x = 0;
y = 0;
z = R * sin(ele * 2 * pi/360);
[sx sy sz] = sphere(30);
hsun = surf(x + 0.5 * sx,y + 0.5 * sy,z + 0.5 * sz,10 * ones(size(sz)));
set(hsun,'LineStyle','none');
hlight = lightangle(0,15);
set(hlight,'Color',[1 1 0.5]);
for n = 0:5:720
lightangle(hlight,n,15);
```

```
view( - 0.1 * n,40 - 40 * n/720);
x = R * cos(ele * 2 * pi/360) * cos((n - 90) * 2 * pi/360);
y = R * cos(ele * 2 * pi/360) * sin((n - 90) * 2 * pi/360);
delete(hsun);
hsun = surf(x + 0.5 * sx,y + 0.5 * sy,z + 0.5 * sz,...
 10 * ones(size(sz)),'linestyle','none');
pause(0.05);
end
```

程序的运行效果如图 12.7 所示。

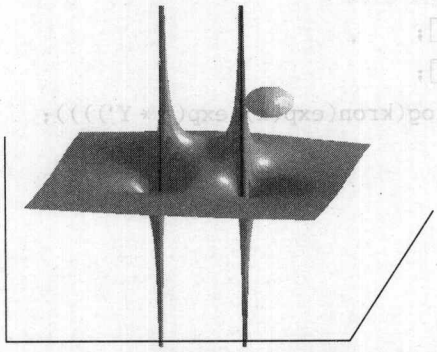


图 12.7 图形的高级控制演示

第五篇

通信系统

第五篇

通信系统

第十三章　傅里叶变换应用于通信系统

本章是信号与系统基本概念和通信工程应用的结合。第一节和第三节举例介绍宽带信号通过低通系统后波形和频谱的变化，解释了理想低通滤波器的不可实现性；第二节举例说明线性延时是保证信号无失真的必要条件；第四节验证因果系统的系数函数实部和虚部之间构成希尔伯特变换对；第五节分别用 MATLAB 编程和 Simulink 仿真的方法实现双边带和单边带的调制与解调；第六节和第七节举例介绍连续时间信号的抽样和 PCM 编码方法。通过本章的例题和讲解，读者可以掌握通信工程中的各个基本要素，为学习“现代通信原理”等后续课程做好充分的准备工作。

本章包括两个 MATLAB 知识点。① 生成常用信号；② 数字显示格式。

学习重点： ① 用 Simulink 仿真实现调制和解调；② 连续时间信号的抽样和 PCM 编码。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
grpdelay	计算群延时	butter	生成巴特沃思滤波器系数
hilbert	希尔伯特变换	conj	取共轭
tripuls	生成三角脉冲	reshape	重新排列矩阵各元素的位置

第一节　利用系统函数 $H(j\omega)$ 求响应

例 13.1（主教材例 5－2 修改）　图 13.1(a)所示 *RC* 低通网络，在输入端

1−1′加入矩形脉冲 $v_1(t)$如图 13.1(b)所示，利用傅里叶分析方法求 2−2′端电压 $v_2(t)$。图中 $E=1$ V，$\tau=0.5$ s。

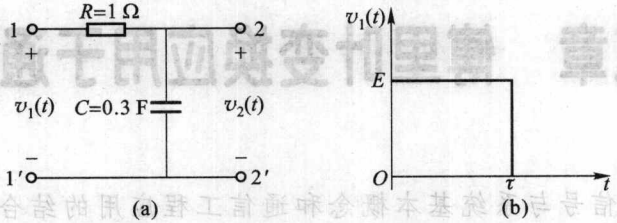


图 13.1 例 13.1 电路图和激励波形

解 定义 $\alpha=\dfrac{1}{RC}$，容易写出低通网络的系统函数 $H(\mathrm{j}\omega)=\dfrac{\alpha}{\alpha+\mathrm{j}\omega}$。下面由理论结果直接写出系统函数 $H(\mathrm{j}\omega)$，然后再乘以激励信号 $v_1(t)$的傅里叶变换 $V_1(\mathrm{j}\omega)$得到响应的频谱 $V_2(\mathrm{j}\omega)$，最后用傅里叶逆变换计算出响应信号 $v_2(t)$。为了说明数值方法的近似性和频谱的意义，本例对定义的系统函数 $H(\mathrm{j}\omega)$求逆变换得到冲激响应 $h(t)$并分析其波形。

§§§MATLAB 文件 ex_13_1.m

```
[t,omg,FT,IFT] = prefourier([-2,2],300,[-50,50],100);
tau = 0.5;                          %激励脉宽 τ
v1 = 0 * t;                         %生成激励信号
v1(t>0&t<tau) = 1;
R = 1;
C = 0.3;
alpha = 1/R/C;
H = alpha./(alpha + j * omg);       %由定义生成系统频响 H(jω)
V1 = FT * v1;                       %变换得到激励信号的频谱 V1(jω)
V2 = V1. * H;                       %计算输出的频谱 V2(jω)
h = IFT * H;                        %逆变换计算系统冲激响应 h(t)
v2 = IFT * V2;                      %逆变换计算输出波形 v2(t)
ex_13_1_plot( );                    %绘制各个信号的波形和频谱
```

绘图结果如图 13.2 所示。可看出由于高频分量被截断了，从系统函数 $H(\mathrm{j}\omega)$逆变换得到的冲激信号有明显的起伏振荡，这是数值方法的欠缺，和理论预测吻合。

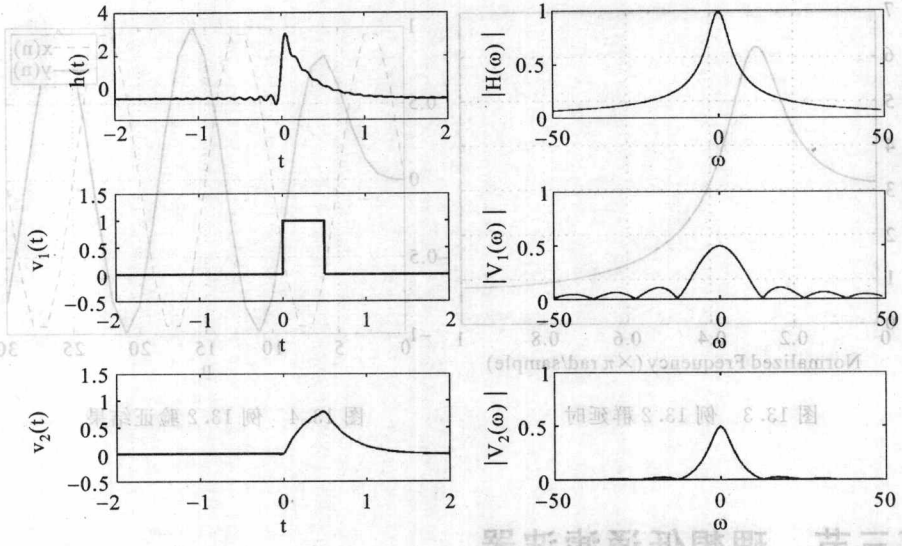


图 13.2　例 13.1 运行结果

第二节　无失真传输

MATLAB 提供了 grpdelay(b,a,N)函数计算系数为 b,a 的数字滤波器的群延时,其中 N 表示待计算的频点数。下面举例说明。

例 13.2　观察 5 阶巴特沃思滤波器的群延时,并选取一个频点进行验证。

解　在命令窗口中输入

```
[b,a] = butter(5,0.3);          %计算 5 阶巴特沃思滤波器的系数
grpdelay(b,a,128);              %绘制群延时
```

这里用 butter 函数产生巴特沃思滤波器的系数。结果如图 13.3 所示,可知巴特沃思滤波器的群延时并非常数,因而它不满足信号传输无失真的条件。同时,当归一化频率为 0.2π 时,可观察到信号延时约为 4 个样点,下面进行验证。

§§§MATLAB 文件 ex_13_2.m

```
n = 0:30;                       %定义序列时间
x = sin(0.2 * pi * n);          %生成测试信号 sin(0.2πn)
y = filter(b,a,x);              %计算输出序列
ex_13_2_plot( );
```

结果如图 13.4 所示,输出比输入大约落后 4 个样点,和群延时计算结果相吻合。

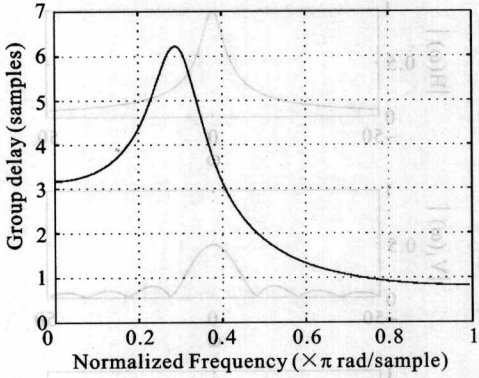


图 13.3 例 13.2 群延时

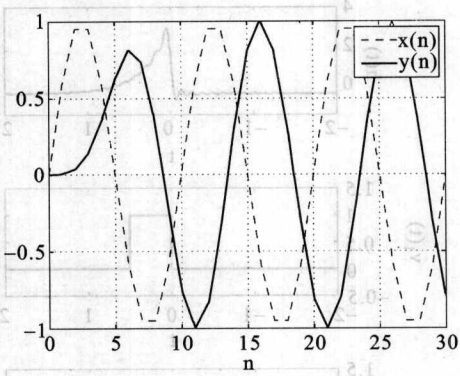


图 13.4 例 13.2 验证结果

第三节 理想低通滤波器

例 13.3(主教材图 5-9 修改) 绘制理想低通滤波器的冲激响应和阶跃响应。

解 首先根据定义生成频率响应,然后由傅里叶逆变换得到冲激响应。

§§§ MATLAB 文件 ex_13_3.m

```
[t,omg,FT,IFT] = prefourier([-3,3],1000,[-15,15],1000);
omgc = 10;                                  %截至频率 ωc
t0 = 1;                                     %线性相位斜率 -t0
H_abs = ((omg<omgc)&(omg>-omgc));           %理想低通滤波器幅频为矩
                                            %形窗
H = H_abs. * exp(-j * t0 * omg);            %理想低通滤波器相频为线
                                            %性相移
h = IFT * H;                                %由频响求冲激响应
ex_13_3_plot(t,h);
```

绘图结果如图 13.5 所示。可见理想低通滤波器的冲激响应是 Sa 函数,中心位置在 $t_0=1$ 处。因为 Sa 函数在 $t<0$ 时非零,因而理想低通滤波器是物理不可实现的。

例 13.4(主教材图 5-13 修改) 绘制矩形脉冲通过不同带宽的理想低通滤波器的响应。

解 首先根据定义生成矩形脉冲的时域波形,然后由傅里叶变换得到其频谱。再乘以由定义得到的理想低通滤波器的频率响应,即可得到输出的频谱,最

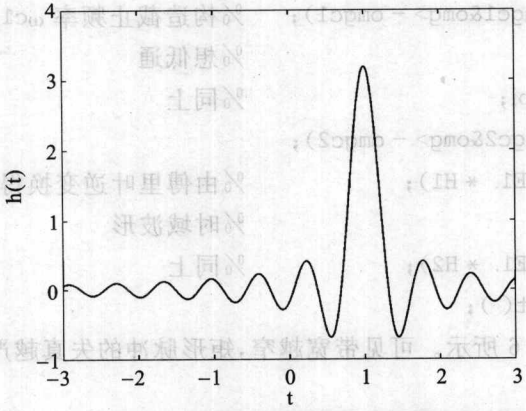


图 13.5　例 13.3 冲激响应波形

后由傅里叶逆变换到时域响应。

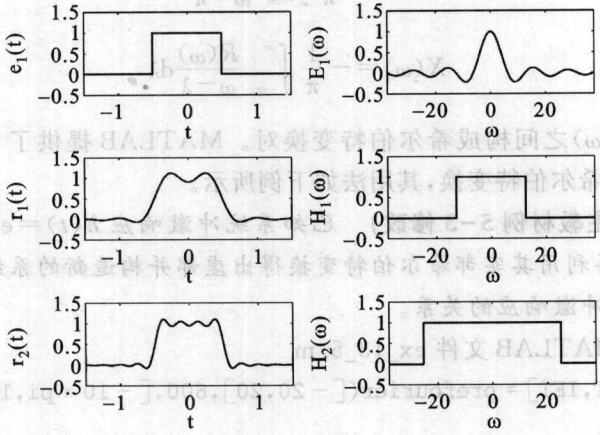


图 13.6　例 13.4 运行结果

§§§MATLAB 文件 ex_13_4.m

```
[t,omg,FT,IFT] = prefourier([ - 1.5,1.5],1000,[ - 12 * pi,12 * pi],
1000);
tau = 1;                          %矩形脉冲宽度 τ=1
e1 = (t> - tau/2&t<tau/2);        %由定义生成矩形脉冲时域波形
E1 = FT * e1;                     %用傅里叶变换计算矩形脉冲的频谱
omgc1 = 4 * pi;                   %理想低通滤波器 1 的截止频率
```

```
H1 = (omg<omgc1&omg> - omgc1);        %构造截止频率 ωc1 相移 0 的理
                                      %想低通
omgc2 = 8 * pi;                       %同上
H2 = (omg<omgc2&omg> - omgc2);
r11 = IFT * (E1. * H1);               %由傅里叶逆变换得到响应 1 的
                                      %时域波形
r12 = IFT * (E1. * H2);               %同上
ex_13_4_plot( );
```

结果如图 13.6 所示。可见带宽越窄,矩形脉冲的失真越严重。

第四节 系统函数的约束特性

一个因果系统的系统函数 $H(j\omega)=R(\omega)+jX(\omega)$,其中

$$R(\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{X(\omega)}{\omega-\lambda}d\lambda \tag{13.1}$$

$$X(\omega)=-\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{R(\omega)}{\omega-\lambda}d\lambda \tag{13.2}$$

即 $R(\omega)$和 $X(\omega)$之间构成希尔伯特变换对。MATLAB 提供了 hilbert 函数实现式(13.1)的希尔伯特变换,其用法如下例所示。

例 13.5(主教材例 5-3 修改) 已知系统冲激响应 $h(t)=e^{-\alpha t}u(t)$,$\alpha=1$。求系统函数,再利用其实部希尔伯特变换得出虚部并构造新的系统函数,验证其因果性及和原冲激响应的关系。

解 §§§MATLAB 文件 ex_13_5.m

```
[t,omg,FT,IFT] = prefourier([ - 20,20],600,[ - 10 * pi,10 * pi],600);
alpha = 1;
h = exp( - alpha * t). * (t> = 0);    %定义因果系统冲激函数 h(t)
H = FT * h;                           %计算系统函数 H(jω)
H1 = conj(hilbert(real(H)));          %用实部希尔伯特变换得出因果的
                                      %系统函数 H1(jω)
h1 = IFT * H1;                        %傅里叶逆变换出冲激函数 h1(t)
ex_13_5_plot( );
```

结果如图 13.7 所示。可见希尔伯特变换得到的系统和原系统相同,只不过因为高频分量被丢弃以及数值计算的误差,在 0 时刻冲激响应跳变处有些振荡。

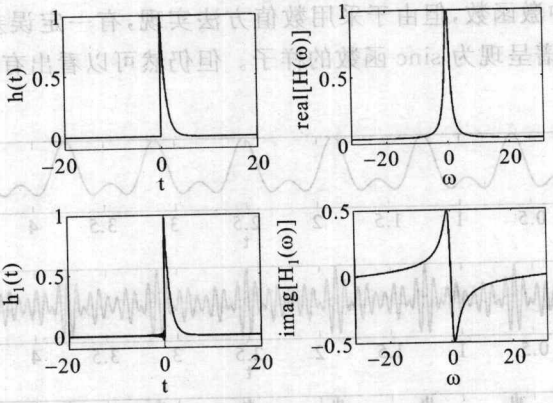


图 13.7　例 13.5 运行结果

第五节　调制与解调

例 13.6(主教材图 5-17(b)和图 5-18(b)修改)　假设基带信号为 $g(t)=3\cos(10t)+2\cos(20t)$，被调制成频带信号 $f(t)=g(t)\cos(100t)$。频带信号在收端又被解调为 $g_0(t)=f(t)\cos(100t)$，并通过低通滤波器

$$H(j\omega)=\begin{cases}1 & |\omega|<30\\ 0 & \text{其他}\end{cases}$$

恢复出基带信号 $g_1(t)$。请绘制上述各个信号的时域波形和频谱。

解　§§§MATLAB 文件 ex_13_6.m

```
[t,omg,FT,IFT] = prefourier([0,5],1000,[-250,250],1000);
g = 3*cos(10*t) + 2*cos(20*t);          %由定义生成基带信号
f = g. *cos(100*t);                     %调制
g0 = f. *cos(100*t);                    %解调
G0 = FT*g0;                             %解调输出的频谱
H = (omg<30&omg>-30);                   %定义低通滤波器 H(jω)
G1 = G0. *H;                            %在频域进行低通滤波
g1 = IFT*G1;                            %逆变换得到时域输出
G = FT*g;                               %计算其他信号的频谱
F = FT*f;
ex_13_6_plot( );
```

绘图结果如图 13.8 和图 13.9 所示。可见解调后成功的恢复出原波形，只不过幅度减小为原来的一半。请注意基带信号只包含两个单频余弦分量，所以

频谱应为四个冲激函数，但由于采用数值方法实现，有一定误差，而且高频分量被截断，因而频谱呈现为 sinc 函数的样子。但仍然可以看出有四个峰值。

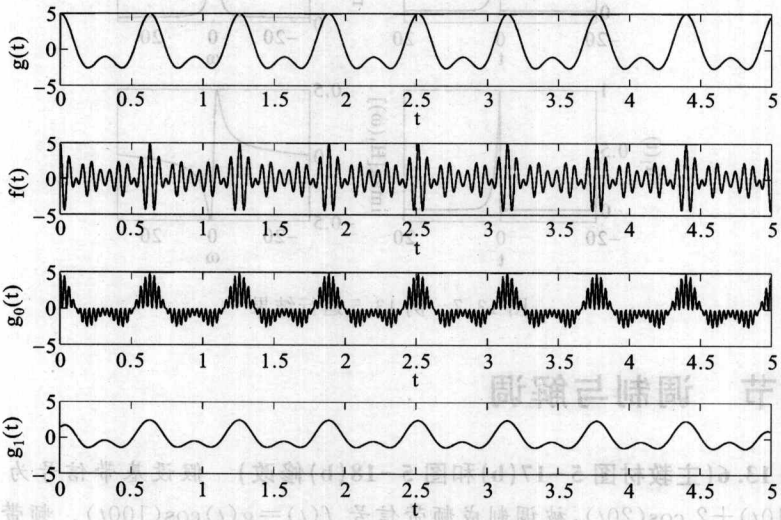


图 13.8 例 13.6 各信号时域波形

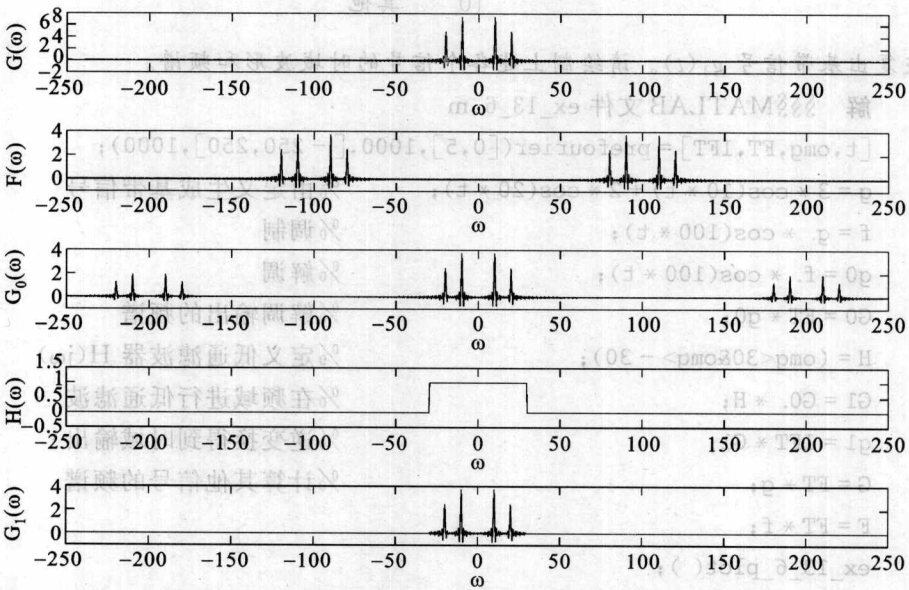


图 13.9 例 13.6 各信号频谱和滤波器频响

MATLAB 提供了强大的仿真工具 Simulink，可以方便地实现调制和解调等功能，举例说明。

例 13.7　基带信号、载波频率和接收端的理想低通滤波带宽都和上例相同。请用 Simulink 实现双边带和单边带的调制/解调。

解　用 Simulink 实现双边带调制和解调的系统结构如图 13.10 所示。

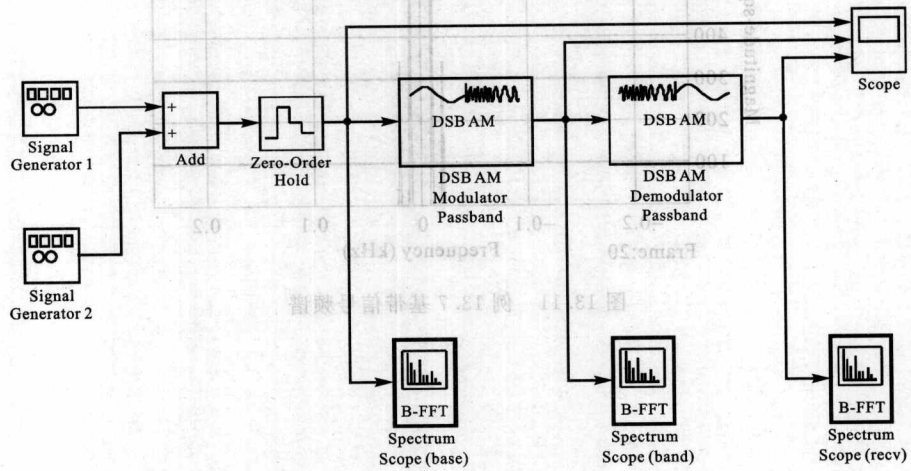


图 13.10　例 13.7 双边带调制/解调框图

仿真结果：基带信号、调制信号和解调后信号的频谱分别如图 13.11、图 13.12 和图 13.13 所示。对比用 MATLAB 编程得到的结果，可见两者基本相同。各信号的波形如图 13.17 所示，同样和编程计算的相似。（注意：编程用的信号是 cos 函数，这里用的是 sin 函数，因而波形上有些差别。）

用 Simulink 实现单边带调制和解调的系统结构如图 13.14 所示。对比双边带框图可见，只是调制解调模块发生变换，其他模块都没有变。

仿真结果中基带信号频谱和双边带的相同，因而不再重复绘出。调制信号和解调后信号的频谱分别如图 13.15 和图 13.16 所示。对比双边带结果，可见调制信号只剩下了上边带。虽然下边带已经被滤掉，但还有些小残余。各信号的波形如图 13.18 所示，由于滤除下边带用到滤波器延时较大，因而可以看到调制和解调信号相比基带信号有明显的延时。

由上实验结果可见，无论是双边带还是单边带，解调信号都较好的恢复了基带信号。

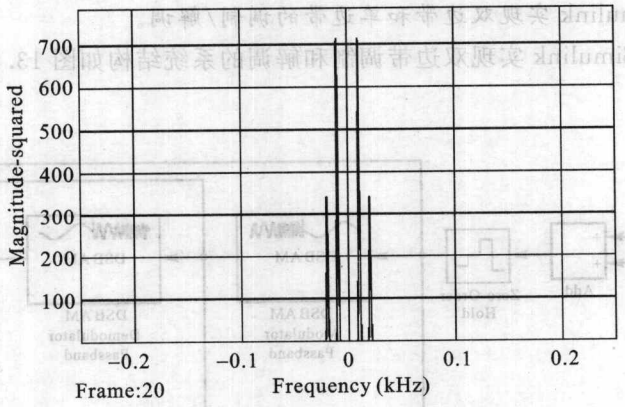


图 13.11 例 13.7 基带信号频谱

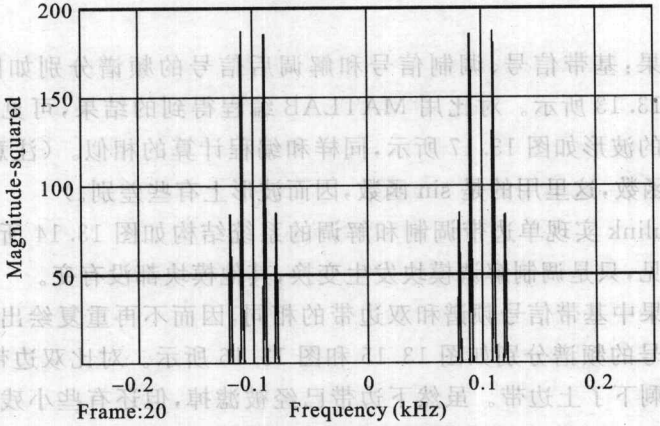


图 13.12 例 13.7 双边带调制信号频谱

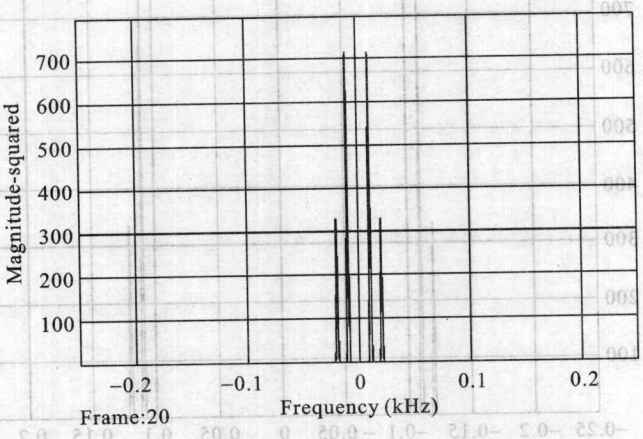


图 13.13　例 13.7 双边带解调后信号频谱

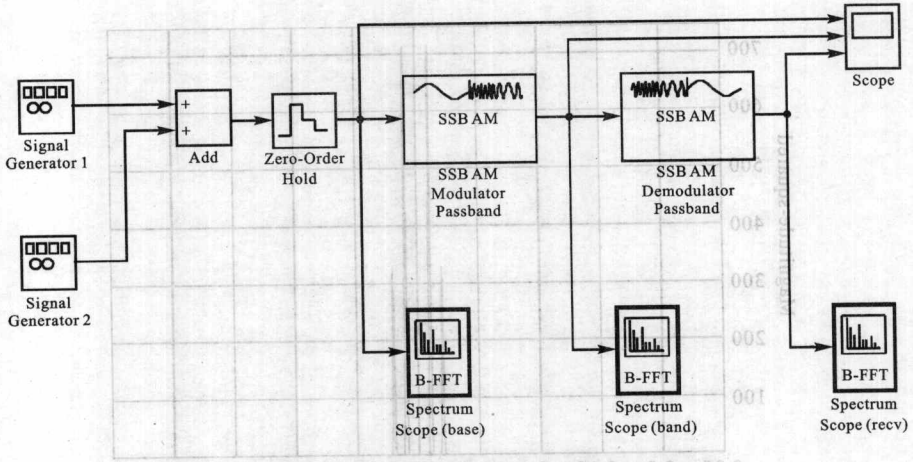


图 13.14　例 13.7 单边带调制/解调框图

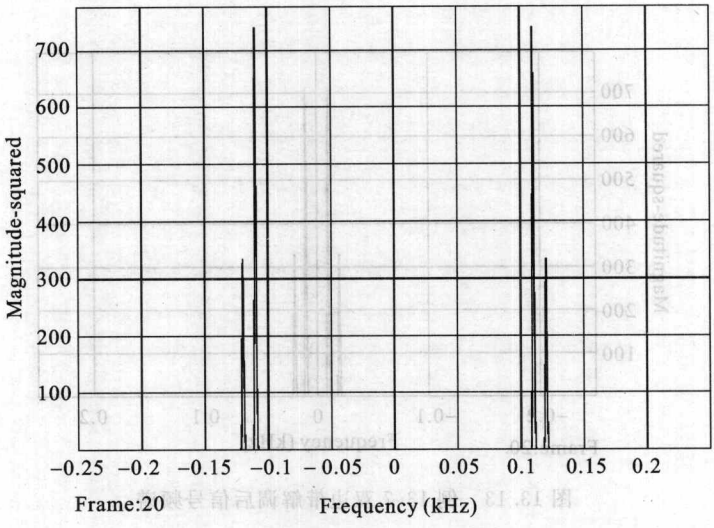


图 13.15 例 13.7 单边带调制信号频谱

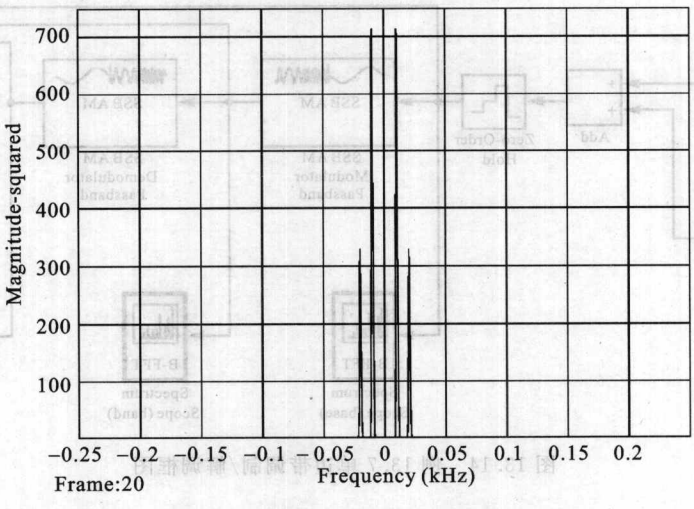


图 13.16 例 13.7 单边带解调后信号频谱

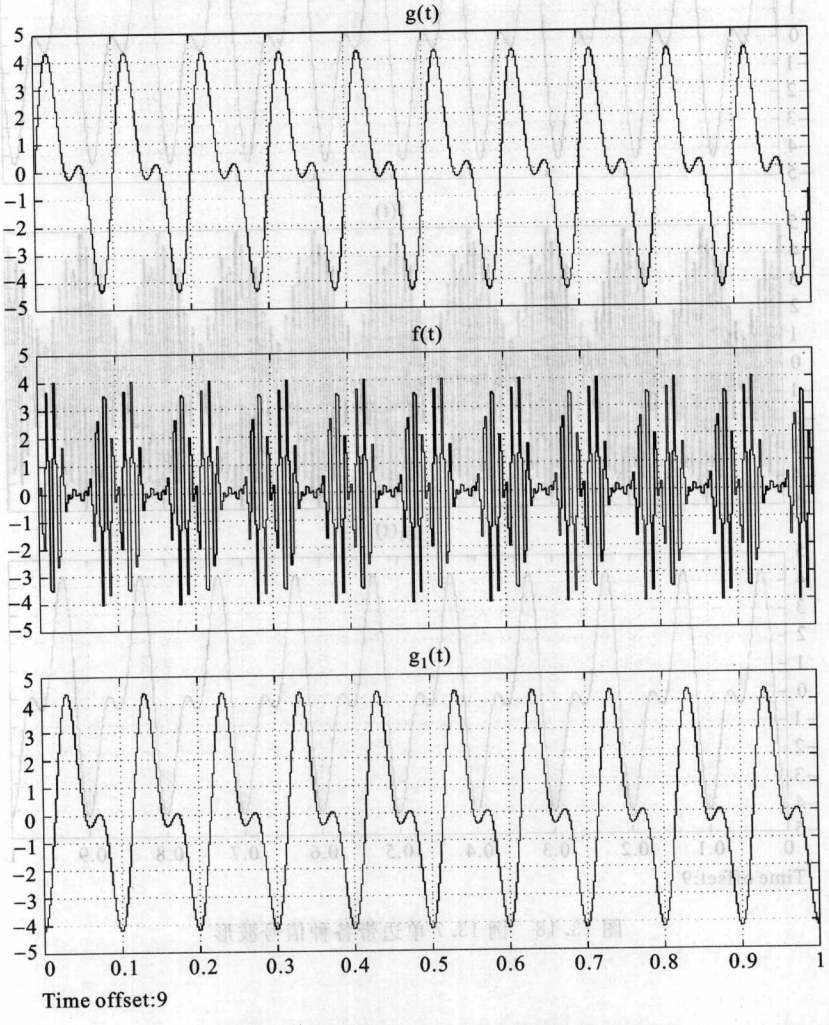


图 13.17　例 13.7 双边带各种信号波形

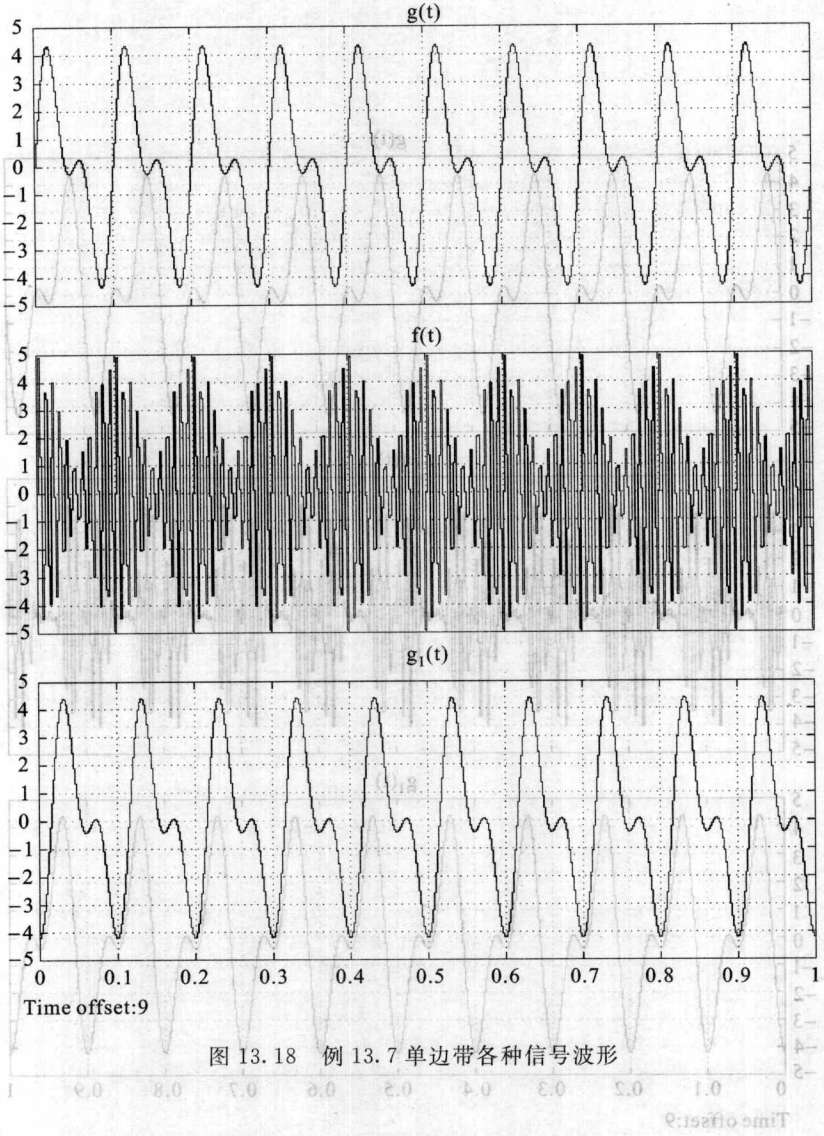


图 13.18 例 13.7 单边带各种信号波形

第六节 从抽样信号恢复连续时间信号

例 13.8 设计信号，编程验证“在时域对连续信号进行抽样将导致频谱周期重复”。

解 计划从两个方向对上述命题进行验证。首先：对频谱进行周期重复，再逆变换回时域，观察是否为原信号的抽样。其次：在时域对原信号进行抽样（注

意 MATLAB 无法实现幅度为无穷大的 $\delta(t)$ 信号，因而用幅度为 1 的单位样值来抽，这样将导致信号能量减小），再将抽样信号变换到频域观察频谱是否呈现周期性。为了让信号的频谱带限从而易于看出周期性，直接在频域定义出三角脉冲作为信号的频谱，再反变换到时域得到波形。MATLAB 提供了 tripuls 函数用于生成周期为 1 且位于零时刻的三角脉冲。

§§§MATLAB 文件 ex_13_8.m

```
T = 60;N = 1000;
OMG1 = 50;
REP = 29;
[t,omg,FT,IFT] = prefourier([ - T/2,T/2],N,[ - REP/2,REP/2],REP *
OMG1);
omgm = pi;
G = tripuls(omg);                    %生成一个三角形的频谱 G(ω)
g = IFT * G;                         %计算该频谱的时域波形 g(t)
temp = 0 * G;
temp(1:OMG1:REP * OMG1) = 1;         %和 G 结构相同的周期为 OMG1 的
                                     %抽样信号 temp
G1 = filter(G(G>0),1,temp);          %通过响应为 G 的非零值的滤波器
                                     %得到重复频谱
g1 = IFT * G1;                       %计算频谱重复后的时域波形 g1(t)
g2 = 0 * g;
g2(1:3 * N/T:N) = g(1:3 * N/T:N);    %对 g(t)以 5 为周期抽样得到 g2(t)
G2 = FT * g2;                        %计算抽样信号的频谱 G2(ω)
ex_13_8_plot;
```

实验结果如图 13.19 所示。首先观察中间两幅图。频谱周期重复后的时域信号表现为重复前的时域信号的抽样，只不过不是冲激信号抽样，而是 sinc 函数的形状，这是因为没有将频谱重复无穷多的周期，而是把高频部分截断的结果。或者说相当于频域加了一个矩形窗，所以时域表现为冲激串和 sinc 函数的卷积，从而把一个个的冲激信号变成了 sinc 函数。另外，由于频谱重复多次，信号能量增大，所以时域抽样信号的幅度增加很多倍。

再看最下面的两幅图，经过时域抽样后，频谱出现周期重复，重复周期等于 2π 除以抽样间隔，这和理论结果完全相同。同时，抽样信号能量衰减很多，因而频谱幅度也变成了原来的几十分之一。

例 13.9（主教材图 5-24）　考察零阶抽样保持后频谱的变化。

解　MATLAB 没有直接完成零阶抽样保持的函数。将首先生成抽样信

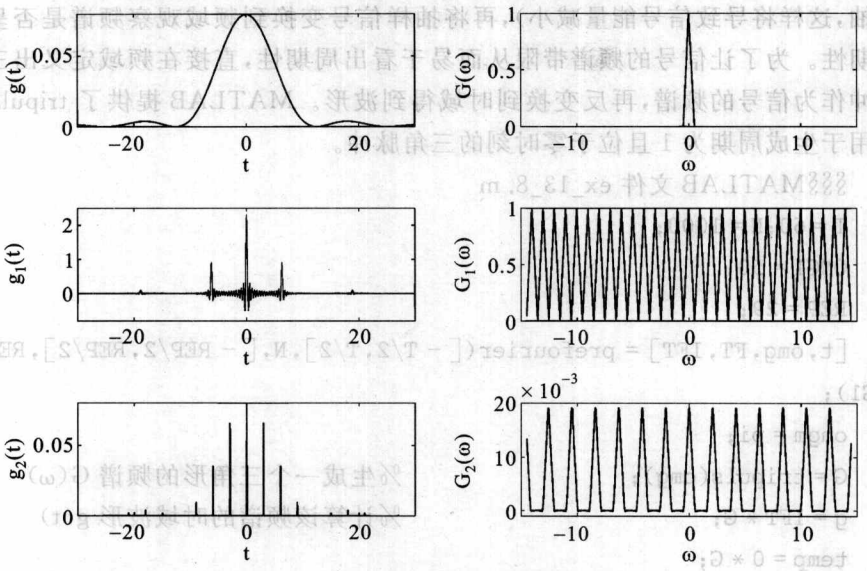


图 13.19 例 13.8 各信号波形和频谱

号，然后通过单位样值响应为矩形序列(序列长度为抽样间隔)的滤波器实现抽样保持。请阅读程序掌握此方法。

```
§§§MATLAB 文件 ex_13_9.m
N = 1000;
KEEP = 50;                          %抽样保持的点数
[t,omg,FT,IFT] = prefourier([-5,5],N,[-10*pi,10*pi],1000);
G = tripuls(omg/5);                 %生成宽度为 5 的三角脉冲
g = IFT*G;                          %逆变换出时域波形
gtemp = 0*g                         %生成和 g 相同结构的全零矢量
                                    %gtemp
gtemp(1:KEEP:N) = g(1:KEEP:N);      %将 gtemp 变成 g 的抽样信号，
                                    %抽样间隔为 KEEP
g1 = filter(ones(1,KEEP),1,gtemp);  %通过 FIR 滤波器，得到抽样保
                                    %持信号
G1 = FT*g1;                         %变换出抽样保持后信号的频谱
ex_13_9_plot( );
```

运行结果如图 13.20 所示。可见和原频谱相比，零阶保持抽样后信号的频谱在低频部分没有变化，而在高频部分多了一些成分，这是由时域出现跳变后引起的。

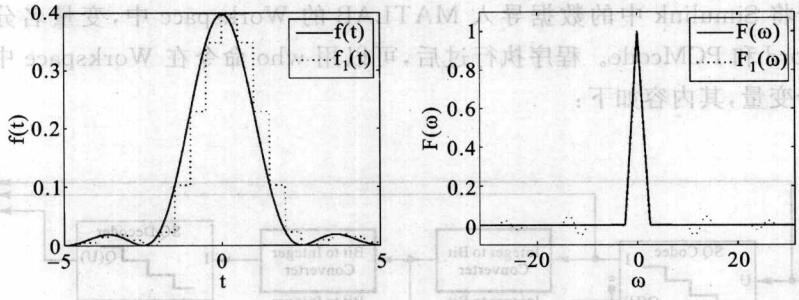


图 13.20　例 13.9 运行结果

MATLAB 知识点(17)——生成常用信号

本节介绍了使用 filter 函数生成周期信号和零阶抽样保持信号的方法。这两种信号也可以用 kron 函数和 reshape 函数组合实现。例如用下面命令将列矢量 x 重复 N 次生成周期序列 xp：

```
xp = kron(ones(N,1),x);
```

若要实现 x 的零阶抽样保持序列 xs，可以先将 x 折叠成 T×N 的矩阵 x1，即

```
x1 = reshape(x,T,N);
```

再将 x1 的第一行(即 x 的抽样)复制 T 行构成矩阵 x2，即

```
x2 = kron(ones(T,1),x1(1,:));
```

最后将 x2“拉直”成列矢量 xs，即抽样保持信号

```
xs = reshape(x2,N * T,1);
```

MATLAB 的信号处理工具箱也提供了很多生成波形的函数，除了生成三角形脉冲的 tripuls 函数，还有生成矩形脉冲的 rectpulse 函数，以及生成周期方波的 square 函数和生成任意波形周期信号的 pulstran 函数，具体请参看 help signal 中 Waveform generation 一段学习。

第七节　脉冲编码调制(PCM)

例 13.10　对一个周期的 $\sin(0.2\pi t)$ 信号用 PCM 进行编码并解码。(参看主教材第五章 5.10 节。)

解 实现 PCM 的系统结构如图 13.21 所示。其中两个 To Workspace 模块用于将 Simulink 中的数据导入 MATLAB 的 Workspace 中，变量名分别为 codeword 和 PCMcode。程序执行过后，可以用 who 命令在 Workspace 中发现这两个变量，其内容如下：

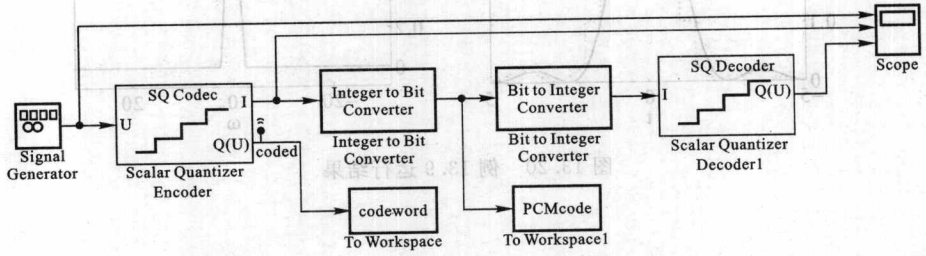


图 13.21 例 13.10 结构框图

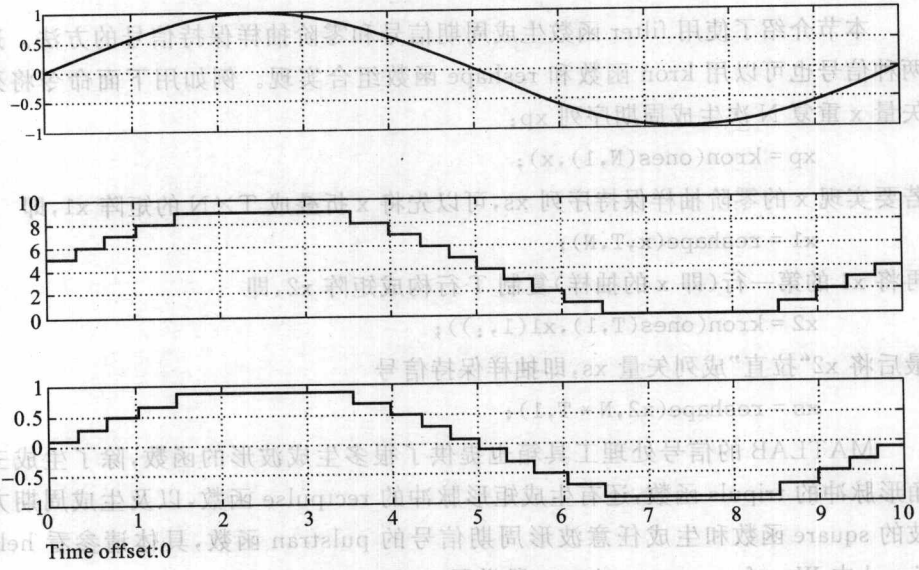


图 13.22 例 13.10 原始和 PCM 编码后的波形

```
codeword =                PCMcode =
    -0.1000                   0     1     0     0
     0.5000                   0     1     1     1
     0.9000                   1     0     0     1
     0.9000                   1     0     0     1
```

0.5000	0	1	1	1
0.1000	0	1	0	1
-0.5000	0	0	1	0
-0.9000	0	0	0	0
-0.9000	0	0	0	0
-0.5000	0	0	1	0
-0.1000	0	1	0	0

就是正弦波量化和编码的结果。

Scope 显示的原始波形、整数编码和编成 PCM 并恢复回原始格式的图形如图 13.22 所示。

MATLAB 知识点(18)——数字显示格式

MATLAB 中所有的数值计算都以双精度浮点格式进行，但默认的显示格式却是 5 位十进制定点数。MATLAB 提供了 format 函数调整数字显示格式，例如 format long 指定以 16 位十进制定点数显示，format short e 以 5 位十进制浮点标准化格式显示，请由 help format 了解其他格式。

在显示数值矩阵时，如果有大于 10^3 或者小于 10^{-3} 的元素，MATLAB 会自动把 10 的整数次幂作为公因子提取出来显示在最前面。如果矩阵中元素大小差别很大，MATLAB 在提取公因子时将优先保证完整显示大元素，这就意味着小元素会显示不出有效数字。例如在命令窗口中输入

```
x = [10000,1000;1,0.001]
```

将回显

```
x =
  1.0e+004 *
    1.0000    0.1000
    0.0001    0.0000
```

如果矩阵很大需要分多屏显示，读者可能会遗漏最前面的公因子而误认为矩阵中有很多零元素，请务必注意防止发生此类错误。

第八节　小结

本章介绍了通信工程中若干利用信号与系统基本概念的知识点。第一至四节首先举例解释了理想低通滤波器、群延时和物理可实现系统等基本概念；第五

节分别用 MATLAB 编程和 Simulink 仿真验证了双边带和单边带的调制和解调方法，让读者初步产生频分复用的基本概念；第六、七节介绍连续时间信号的抽样、编码和恢复，使读者初步了解数字通信系统的原理与特点。调制和抽样理论最重要的贡献是构成了频分复用与时分复用通信系统，这两种多路复用技术将在第十五章的通信系统仿真中作为综合练习供读者进一步实践。

练习题

1. 已知有两个基带信号 $x(t)=\cos t$ 和 $y(t)=2\cos(2t)$，分别用载波 $c_1(t)=\cos(100t)$ 和 $c_2(t)=\cos(110t)$ 进行传输。试用 Simulink 仿真此通信系统，设计低通滤波器参数，并绘制各个关键点的信号波形。

2. 已知连续时间信号 $x(t)=\sin(100\pi t)+\cos(200\pi t)$，以 400 Hz 速率对 $x(t)$ 进行抽取得到离散信号 $x_1(n)$，再对 $x_1(n)$ 依次进行 8 比特量化和 PCM 编码后分别得到量化信号 $x_2(n)$ 和序列 $x_3(n)$，试用 Simulink 仿真产生并绘制上述各个信号。若要由 $x_3(n)$ 恢复出连续时间信号 $y(t)$，请设计低通滤波器并绘制 $y(t)$。

第十四章　信号的矢量空间分析

主教材的第六章有两个目的：一是傅里叶变换在理论上的提高，由内积空间的概念导出一般正交函数展开；二是进一步介绍相关和匹配滤波器等在通信工程中的应用。作为配套实验教材，本章只侧重于主教材中工程应用的部分。第一节举例介绍用 MATLAB 实现相关运算的方法；第二节展示功率信号的相关函数和功率谱之间的关系；第三节以 *RC* 低通网络降噪为例介绍信号通过线性系统的分析方法；第四节以测距系统的匹配滤波器接收为例解释通信和雷达系统对脉冲波形的要求。

本章包括一个 MATLAB 知识点：图形窗口中显示数学符号。

学习重点：MATLAB 实现相关运算的方法。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
xcorr	实现离散相关运算	randn	产生高斯分布的随机序列
psd	计算功率谱密度函数		

第一节　相关

由相关的定义式，它和卷积非常相似，而且省掉了移位积分前翻转的步骤。因而可以利用前面卷积的数值计算方法直接得到相关运算的数值方法

$$f(m\Delta t)\approx\Delta t\sum_{n} f_1(n\Delta t)f_2(n\Delta t-m\Delta t)$$

而 MATLAB 提供了 c＝xcorr(a,b)函数实现

$$c(K+m)=\sum_{n=1}^{K} a(n)b(n-m)$$

的离散相关运算，其中 $-K<m<K$，$K=\max(L_a,L_b)$，L_a 和 L_b 分别表示 a、b 的长度。对比以上两式，可发现很容易用 xcorr 函数实现连续函数的相关。

例 14.1(主教材图 6-11 修改) 比较卷积和相关运算，并验证相关定理。

解 分别从时域计算矩形脉冲和三角脉冲的卷积和相关，然后从变换域利用卷积定理和相关定理计算。最后绘图验证两种方法得到的计算结果。

§§§MATLAB 文件 ex_14_1.m

```
T = 4;                                          %时域抽样时间范围
N = 1 000;                                      %时域抽样点数
[t,omg,FT,IFT] = prefourier([ - T/2,T/2],N,[ - 16 * pi,16 * pi],
1 000);
f1 = (t> = 0&t<1);                              %定义矩形脉冲 f1 和三角脉冲 f2
f2 = (t> = 0&t<1). * (1 - t);
xtemp = T/N * conv(f1,f2);                      %做卷积并从适当位置截取
x = xtemp(N/2:N/2 + N - 1);
ytemp = T/N * xcorr(f1,f2);                     %做相关并从适当位置截取
y = ytemp(N/2:N/2 + N - 1);
x1 = IFT * ((FT * f1). * (FT * f2));            %利用卷积定理求卷积
y1 = IFT * ((FT * f1). * conj(FT * f2));        %利用相关定理求相关
ex_14_1_plot();
```

程序结果如图 14.1 所示。可见用 MATLAB 求卷积和相关的步骤很相似，而且用卷积/相关定理从频域求卷积/相关的结果和时域完全相同。

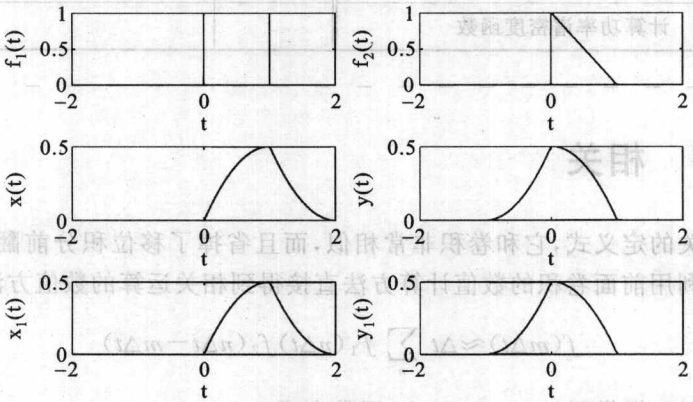


图 14.1 例 14.1 运行结果

第二节　能量谱和功率谱

功率谱和白噪声的概念在“随机过程”和“现代通信原理”等后续课程中有重要应用。本课程仅给出定义和解释。MATLAB 中可以用 randn 函数生成高斯分布的随机序列，用 psd 估计功率谱密度。

例 14.2　计算白噪声的自相关函数和功率谱密度。

解　§§§MATLAB 文件 ex_14_2.m

```
N = 100 000;
w = randn(N,1);          %生成随机信号
R = xcorr(w);            %计算自相关函数
P = psd(w);              %计算功率谱密度函数
ex_14_2_plot();
```

实验结果如图 14.2 所示。取十万个样点，其自相关函数基本呈冲激函数状，功率谱密度也表现为一条直线，说明噪声功率在各个频段是同分布的，即白色噪声。

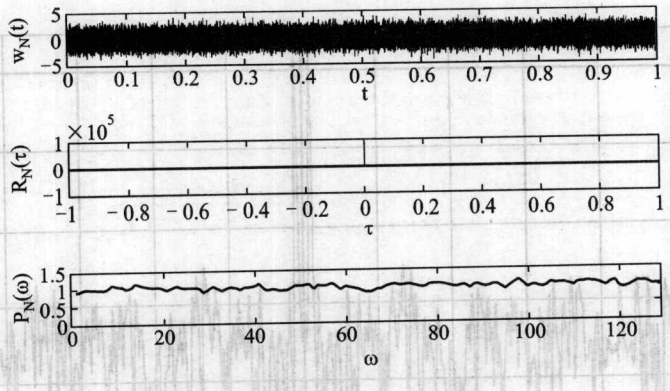


图 14.2　例 14.2 运行结果

第三节　信号通过线性系统的分析

例 14.3(主教材例 6-5 和例 6-6 修改)　正弦信号 $e(t)=\sin t$ 混入了功率谱密度为 1 的白噪声，将混合信号通过如图 14.3 所示的 RC 低通网络，求输入和输出信号的功率谱。

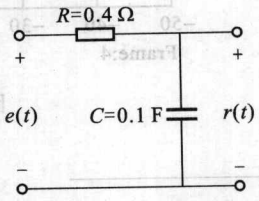


图 14.3　例 14.3RC 低通网络

解　已知 RC 低通网络的传递函数为

$$H(s)=\frac{\frac{1}{RC}}{s+\frac{1}{RC}}=\frac{25}{s+25} \tag{14.1}$$

根据题意，用 Simulink 绘出系统框结构如图 14.4 所示[①]。

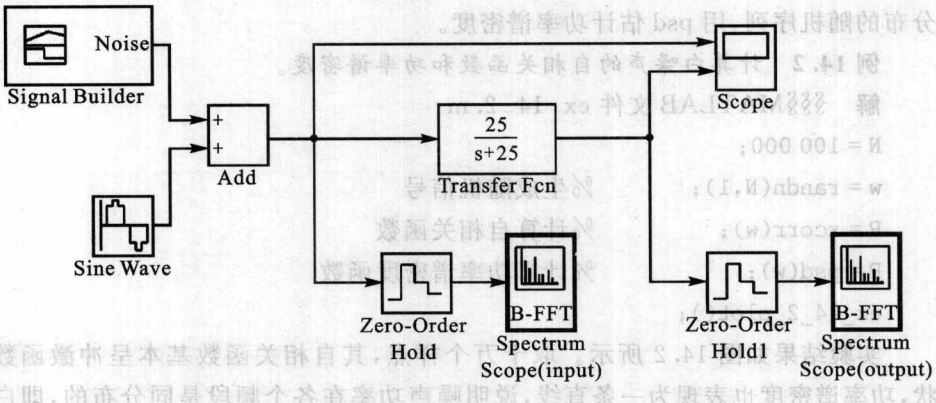


图 14.4 例 14.3 系统结构框图

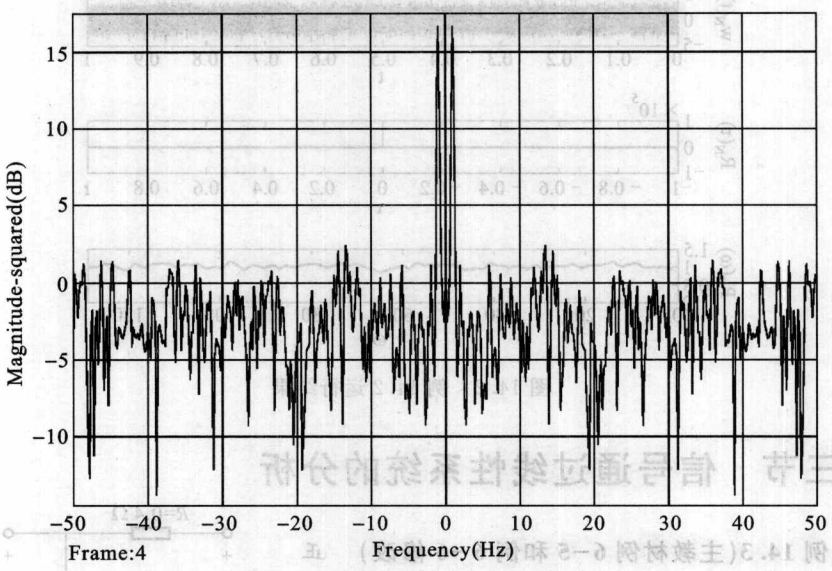


图 14.5 例 14.3 输入信号功率谱

① 请注意 zero order holder 的抽样频率一定要和生成 sin 信号的抽样频率一致。注意设置 sin 信号频率的单位为 rad/s。

运行程序后，输入和输出信号的功率谱分别如图 14.5 和图 14.6 所示。可见混有噪声的单频信号经过低通网络后，输出信号中的噪声成分，尤其是高频分量已经得到了很大程度的衰减。这个结论在时域波形上看得也很明显，如图 14.7 所示，输出信号明显呈正弦模样，并且相对输入平滑许多。

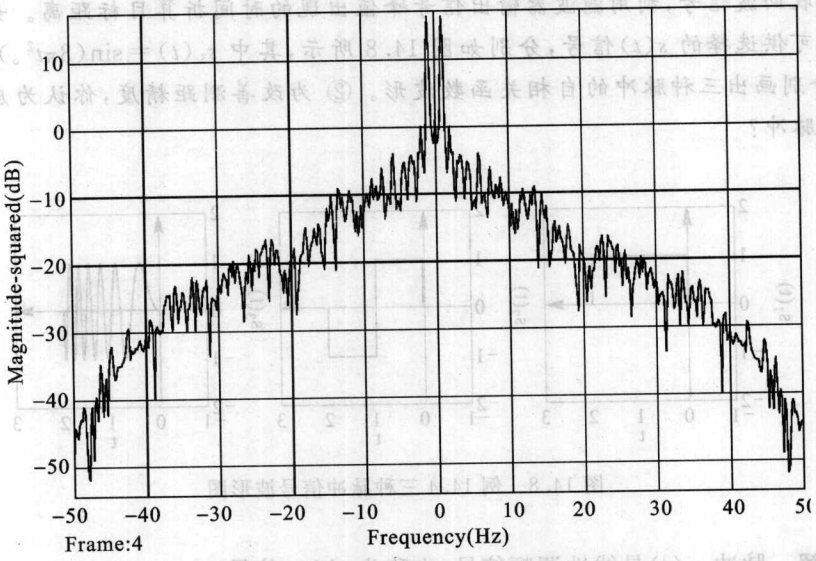


图 14.6　例 14.3 输出信号功率谱

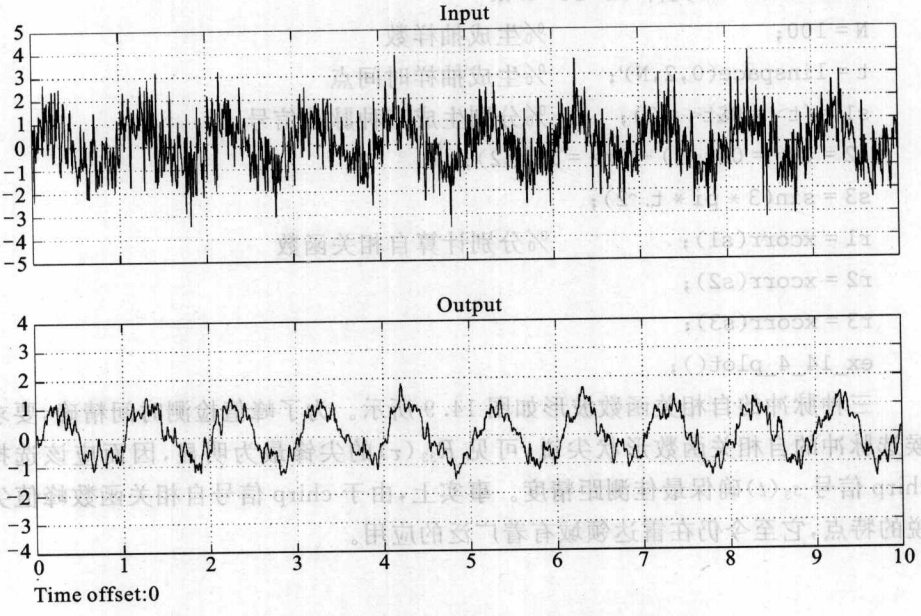


图 14.7　例 14.3 输入和输出信号波形

第四节 匹配滤波器

例 14.4(主教材例 6-7 修改) 在测距系统中,发送信号 $s(t)$,以匹配滤波器接收回波信号,利用滤波器输出信号峰值出现的时间折算目标距离。如果有三种可供选择的 $s(t)$ 信号,分别如图 14.8 所示,其中 $s_3(t)=\sin(3\pi t^2)$。求:① 分别画出三种脉冲的自相关函数波形。② 为改善测距精度,你认为应选用哪种脉冲?

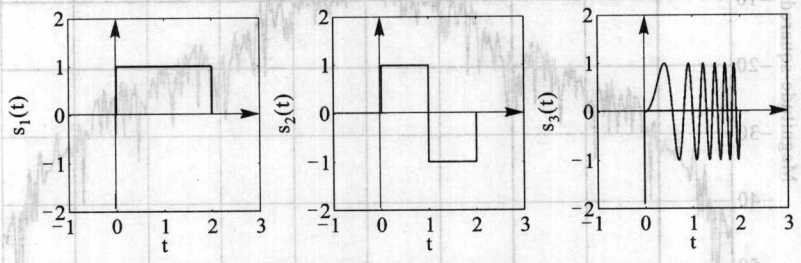


图 14.8 例 14.4 三种脉冲信号波形图

解 脉冲 $s_3(t)$ 是线性调频信号,也称为 chirp 信号。

§§§MATLAB 文件 ex_14_4.m

```
N = 100;                    %生成抽样数
t = linspace(0,2,N)';       %生成抽样时间点
s1 = (t> = 0&t< = 2);       %分别生成三种脉冲信号
s2 = (t> = 0&t<1) - (t> = 1&t<2);
s3 = sin(3 * pi * t.^2);
r1 = xcorr(s1);             %分别计算自相关函数
r2 = xcorr(s2);
r3 = xcorr(s3);
ex_14_4_plot();
```

三种脉冲的自相关函数波形如图 14.9 所示。为了峰值检测时间精确,要求候选脉冲的自相关函数形状尖锐,可见 $R_{33}(\tau)$ 的尖锋最为明显,因而应该选择 chirp 信号 $s_3(t)$ 确保最佳测距精度。事实上,由于 chirp 信号自相关函数峰值尖锐的特点,它至今仍在雷达领域有着广泛的应用。

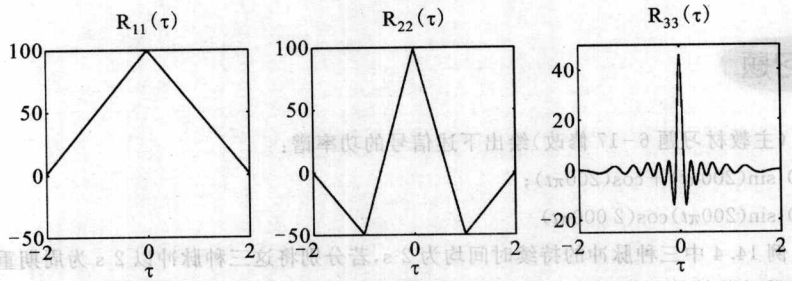


图 14.9　例 14.4 三种脉冲的自相关函数波形

MATLAB 知识点(19)——图形窗口中显示数学符号

如果阅读过中国高校电工电子课程网上和例题配套的绘图程序，读者可能发现程序输出的文本和图形窗口中显示的符号大不相同，例如例 14.4 配套的“ex_14_4_plot.m”文件中为图 14.9 的第一个子图添加标题的语句是

```
title('R_{11}(\tau)');
```

但该子图显示的标题却是 $R_{11}(\tau)$！

为了在图形窗口中显示数学符号，MATLAB 支持TEX/LATEX语法中与数学符号相关的内容，常用的有：花括号{}表示功能作用域和限定域，下画线_表示其后的内容为下标，尖号^表示其后的内容为上标，反斜线\表示特殊符号，\alpha、\beta、\gamma、…分别表示希腊字母 α、β、γ 等。此外，MATLAB 还提供了 latex 函数把符号变量转换为LATEX字符串。

感兴趣的读者可以自学LATEX的相关著作，推荐 Tobias Oetiker 等著，中国CTEX用户小组翻译的《一份不太简短的LATEX2ε介绍》(http://bbs.ctex.org 免费下载)。

第五节　小结

本章的重点是相关运算和匹配滤波器接收。第一节和第四节分别举例介绍了相关运算和匹配滤波器的 MATLAB 实现方法。此外，相关和正交等概念是码分复用技术的理论基础，请读者参考主教材深入学习。下一章中，码分复用技术将和频分复用与时分复用一起，为读者演示出丰富多彩的通信系统。

练习题

1.（主教材习题 6-17 修改）绘出下述信号的功率谱：

（1）$\sin(200\pi t)+\cos(200\pi t)$；

（2）$\sin(200\pi t)\cos(2\,000\pi t)$

2. 例 14.4 中三种脉冲的持续时间均为 2 s，若分别将这三种脉冲以 2 s 为周期重复两个周期（新脉冲的持续时间均为 4 s），重做该例题，并和对应原脉冲的测距精度进行比较，说明重复后的测距精度是否有提高。

第十五章　通信系统仿真

通信工程是信号与系统基本概念的典型应用。本章将仿真验证现代通信工程中的三种主要复用方式：频分复用、时分复用和码分复用。通过本章的仿真练习，读者将对数字通信系统和现代通信理论的主要内容有初步了解。

第一节　背景知识

15.1.1　频分多址(FDMA)

(参看主教材第五章 5.11 节)频分多址是将通信的频段划分成若干信道频率范围，每对通信的设备工作在某个特定的频率范围内，即不同的通信用户是靠不同的频率划分来实现通信的。早期的无线通信系统，包括现在的无线电广播、短波通信、大多数专用通信网都是采用频分多址技术来实现的。

15.1.2　时分多址(TDMA)

(参看主教材第五章 5.11 节)时分多址是将通信信道在时间坐标上划分成若干等间隔的时隙，每对通信设备将工作在某个分配的或者是指定的时隙上，不同的通信用户是靠不同的时隙划分来实现通信的，现在的数字蜂窝无线通信系统 GSM 就采用了时分多址技术。

15.1.3　码分多址(CDMA)

(参看主教材第六章 6.11 节)码分多址是利用码组的正交性，将承载的不同用户的通信信息区分开来。每对通信设备工作在某个分配的码组实现通信。现在的数字蜂窝无线通信 CDMA，第三代移动通信系统 WCDMA、CDMA2000、SC-CDMA 都采用了码分多址技术。码分多址要求通信的码组之间有很好的

正交性。

有一种获取正交码组的方法是利用M序列发生器。M序列是最大长度线性反馈移位寄存器序列的简称。M序列发生器的结构图如15.1所示，其中 a_i 表示各个寄存器的状态，c_i 可取0或者1。

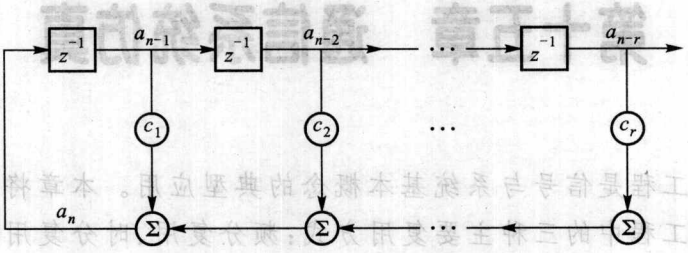


图15.1 M序列发生器的原理框图

$$F(x)=\sum_{i=0}^{r} c_i x^i \tag{15.1}$$

式(15.1)是关于 x 的多项式，系数 c_i 表示了序列生成器的反馈连线的特征，称为移位寄存器生成器函数的特征多项式。

由于 r 位二进制移位寄存器最多可以取 2^r 个不同的状态，因此每个移位寄存器序列最终都是周期序列，并且其周期 $n \leqslant 2^r$。M序列具有很强的自相关特性和很弱的互相关特性，周期为 2^r-1 的M序列可以提供 2^r-1 个正交码组。

第二节 通信系统仿真综合实验

15.2.1 FDMA的Simulink仿真

如图15.2所示是一个FDMA系统的仿真模型，其中部分模块的参数设置如表15.1至表15.6所示。

表15.1 Signal Generator类模块参数设置

	Wave form	Amplitude	Frequency(Hertz)
Signal Generator 1	Sine	1	4
Signal Generator 2	Square	1	0.5
Signal Generator 3	Saw tooth	1	1

表 15.2　DSB AM Modulator Passband 类模块参数设置

	Input offset	Carrier frequency	Initial phase
DSB AM Modulator Passband 1	1	40	0
DSB AM Modulator Passband 2	1	60	0
DSB AM Modulator Passband 3	1	80	0

表 15.3　AWGN Channel 类模块参数设置

	Initial seed	Mode	Variance
AWGN Channel	67	Variance from mask	0.01

表 15.4　Zero-order holder 类模块参数设置

	Sample time
Zero-order holder 1	0.001
Zero-order holder 2	0.001
Zero-order holder 3	0.001

表 15.5　DSB AM Demodulator Passband 类模块参数设置

	Lowpass filter	Filter order	Cutoff frequency
DSB AM Demodulator Passband 1	Butterworth	4	20
DSB AM Demodulator Passband 2	Butterworth	4	20
DSB AM Demodulator Passband 3	Butterworth	4	20

表 15.6　Scope 类模块参数设置

	Number of axes	Time range	Sampling
Scope 1	auto	Sample time	0.01
Scope 2	auto	Sample time	0.01
Scope 3	auto	Sample time	0.01

(1) 利用 Simulink 中的相应模块，搭建图 15.2 所示的系统仿真框图，并设置相应的参数。

(2) 图 15.2 中的六个 Analog Filter Design 滤波器的作用分别是什么？根据已知的参数设置它们的参数，然后进行系统仿真，记录下三个 Scope 中显示的波形。

(3) 尝试用 Spectrum Scope 对各信号进行频谱分析[①]。

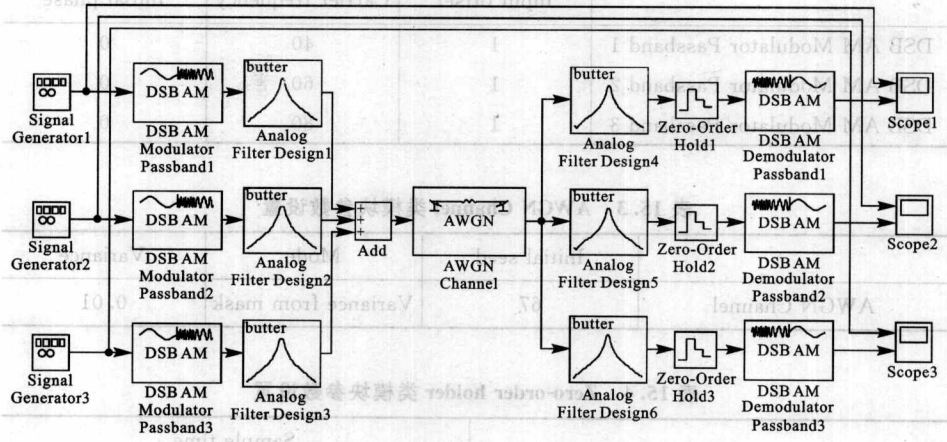


图 15.2 FDMA 的 Simulink 仿真框图

15.2.2 TDMA 的 Simulink 仿真

如图 15.3 所示是一个 TDMA 系统的仿真框图,其中三个 Signal Generator 的设置与上一题相同,Multiplex 和 Demultiplex 是时分多址复用/解复用单元,需要读者来完成。

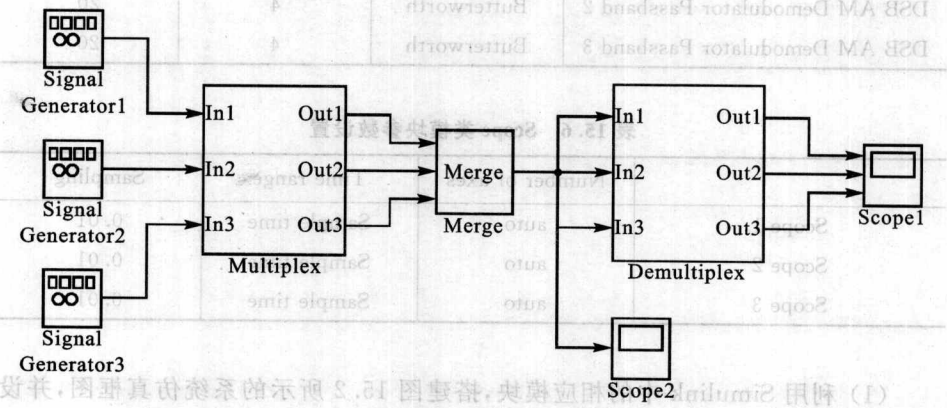


图 15.3 TDMA 的 Simulink 仿真框图

① 连续信号用 Spectrum Scope 分析时需通过 Zero-order Holder 抽样,将 Spectrum Scope 的 buffer size 设成 512,buffer overlap 设成 256。

(1) 设计时分多址复用单元 Multiplex，其中可能用到的模块在表 15.7 中列出。

表 15.7　复用单元 Multiplex 所需模块和用途说明

模　块　名	用　　途
Pulse generator	产生占空比为 1/3 的方波脉冲信号
Zero-order hold	抽样保持
Unit delay	将 Pulse generator 产生的信号分别延时 1 个和 2 个方波宽度，作为其他两路的选通信号
Enabled subsystem	利用三路时间上错开的方波使能，将三路输入输出分配到不同时隙

(2) 时分多址接收单元 Demultiplex 与发送单元是什么关系？在实际中，要使接收端能够还原出三路信号，接收端由 Pulse generator 产生的门控脉冲与发送端的脉冲有什么关系？

(3) 完成该系统，记录 Scope 显示的波形。

15.2.3　CDMA 的 Simulink 仿真

(1) 设计一个简单的 CDMA 通信系统的仿真模型，图 15.4 供参考，要点如下：

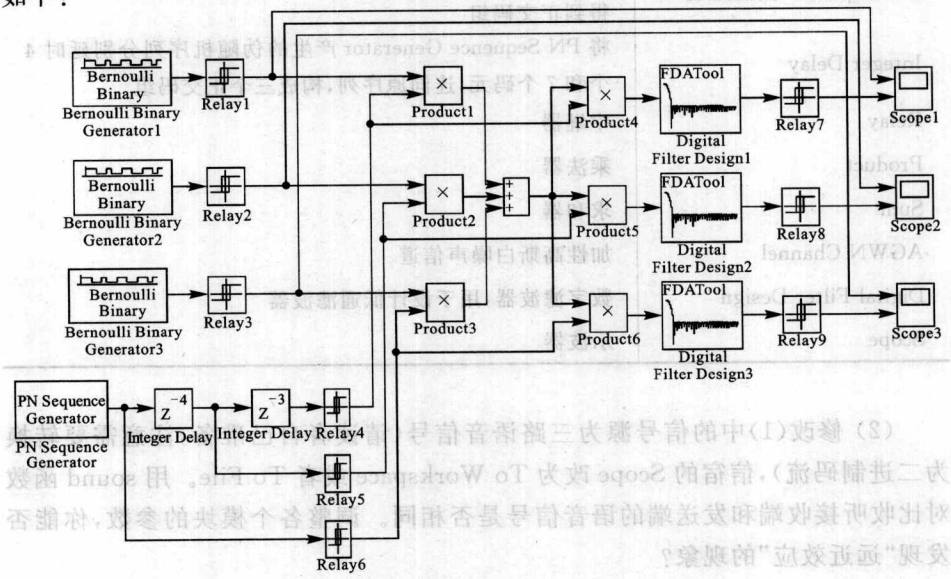


图 15.4　CDMA 的 Simulink 仿真框图(供参考)

• 传输信号为三路二进制伯努利随机序列(称为基带信号)。

• 三个正交码组由 M 序列发生器生成的序列及其延时 4 个和 7 个码元的序列构成,码元宽度为基带信号码元宽度的 1/50。

• 基带信号与正交码组都需利用中继器(施密特触发器)将单极性信号转换成双极性信号。

• 将每一路基带信号与正交码组相乘(称为直接扩频),然后将三路信号相加后通过 AWGN 信道进行传输。

• 利用正交信号强自相关性和弱互相关性,在接收端用三路正交信号分别与接收信号相乘来提取每一路发送的信号。

• 接收端每一路提取的信号可通过一个低通滤波器来滤除高频分量,并利用中继器(施密特触发器)转换成二进制双极性信号。

要求:

• 设计出该系统的仿真框图,注明主要参数。

• 记录比较发送端和接收端各自的波形。

可能用到的模块在表 15.8 中列出。

表 15.8 CDMA 仿真所需模块和用途说明

模 块 名	用 途
Bernouli Binary Generator	产生二进制伯努利随机序列
PN Sequence Generator	产生由 M 序列发生器生成的伪随机序列,经延时后可以得到正交码组
Integer Delay	将 PN Sequence Generator 产生的伪随机序列分别延时 4 个和 7 个码元,连同原序列,构成三个正交码组
Relay	中继器
Product	乘法器
Sum	求和器
AGWN Channel	加性高斯白噪声信道
Digital Filter Design	数字滤波器,用于设计低通滤波器
Scope	示波器

(2) 修改(1)中的信号源为三路语音信号(请读者自己准备,注意需要转换为二进制码流),信宿的 Scope 改为 To Workspace 或者 To File。用 sound 函数对比收听接收端和发送端的语音信号是否相同。调整各个模块的参数,你能否发现“远近效应”的现象?

15.2.4　三种多址方式的比较

现在已经实现了三种多址访问的通信方式，请简要评价这三种方式的性能，包括频带资源占用、抗噪声能力、链路数量等。

15.2.4 三种全址方式的比较

表给出了三种全址访问的通信方式，通常要评价这三种方式的性能，
应需硬件资源占用，指令执行效率，编码复杂度等。

第六篇

控制系统

第六篇

控制系统

第十六章　反馈系统

本章首先完整地介绍线性系统的四种描述方法及其属性，其中传递函数模型和状态空间模型在第三章中已有初步讲解；第二节举例介绍反馈系统的两个基本功能：改善系统频响特性和使不稳定系统成为稳定系统；第三节介绍根轨迹的作用及其绘制方法。

本章包括三个 MATLAB 知识点。① 函数和运算符重载；② 单元数组；③ 用户交互操作。

学习重点：反馈系统的基本特性及用 MATLAB 实现和验证的方法。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
zpk	建立零点-极点-增益模型	frd	建立频率响应数据模型
ltiprops	查看 LTI 模型属性	ssdata	获取状态空间模型的属性
series	建立串联的 LTI 模型	parallel	建立并联的 LTI 模型
feedback	建立带反馈的 LTI 模型	connect	建立任意组合的 LTI 模型
rlocus	绘制根轨迹	rlocfind	交互式计算根轨迹上的增益
num2str	将数字型转为字符串型		

第一节　引言

16.1.1　控制系统工具箱中的 LTI 模型

MATLAB 的控制系统工具箱(Control System Toolbox)提供了四种 LTI 模型，如表 16.1 所示。

表 16.1 控制系统工具箱中的四种 LTI 模型及其定义方法

名称	意义	定义	参数
tf 模型	传递函数模型	tf(num,den,…)	传递函数的分子和分母系数
zpk 模型	零点-极点-增益模型	zpk(z,p,k,…)	零点、极点和增益
ss 模型	状态空间模型	ss(A,B,C,D,…)	状态方程和输出方程的系数矩阵
frd 模型	频率响应数据模型	frd(resp,freq,…)	频率响应和频率抽样点

可以通过调用 ltimodels('tf')等命令查看上述模型的详细定义和说明。除了直接定义,上述部分模型之间还可以相互转换,使用方式请参见下例。另外,MATLAB 还支持独立的转换命令包括 tf2zpk、tf2zp、tf2ss、ss2zp 和 ss2tf 等。

对于一个给定的模型,MATLAB 提供了 class(sys)函数返回其模型类型,提供了 isa(sys,'tf')函数判断其是否为某种(例如'tf')模型。

例 16.1 求下述系统的零、极点

$$H(s)=\frac{s^2-2s+1}{3s^3+s^2-6s+1}$$

解 在命令窗口中输入

```
b=[1 -2 1];
a=[3 1 -6 1];
sys=tf(b,a);
sys1=zpk(sys)
```

运行结果为

```
Zero/pole/gain:
   0.33333(s-1)\^{}2
-----------------
(s+1.66)(s-1.152)(s-0.1744)
```

16.1.2 访问 LTI 模型的属性

LTI 模型包括公共属性和私有属性,其中公共属性是所有 LTI 模型都拥有的,包括抽样时间、输入/输出延时和通道名称等,读者可键入 ltiprops 自己学习。LTI 模型的私有属性主要包括定义该模型的必要参数等,例如 tf 模型的私有属性如表 16.2 所示,读者可以键入 ltiprops('tf')详细查看。其他模型的属性也都很直观,不再分别介绍。

标准的访问模型属性的方法是 get 和 set,另外,每个模型还提供了直接获取其所有属性的函数,如 ssdata。以 tf 模型为例,这些函数如表 16.3 所示。对其他模型,使用方法类似,读者可自行验证。

表 16.2　'tf'模型的私有属性

名　称	意　义	内　容
num	传递函数分子系数	行矢量(对单输入系统)
den	传递函数分母系数	行矢量(对单输出系统)
Variable	变量名	包括's'、'p'、'z'、'z^-1'和'q'等

表 16.3　访问模型属性的函数(以'tf'模型为例)

名称和使用方法	说　明
x=get(sys,'num')	得到 sys 的分子多项式系数并保存在 x 中
set(sys,'num',x,…)	将 x 设置为 sys 的分子多项式系数
[num,den]=tfdata(sys)	读取 sys 的分子和分母多项式系数并赋值到 num 和 den 中

最后,还可以通过直接访问单元数组或结构中元素的方法访问模型的属性,单元数组将在本章第二节后的 MATLAB 知识点中单独介绍。

例 16.2　对上例系统,求出其用状态方程描述的 A 和 B 矩阵。

解　接上例命令行,在命令窗口中输入

```
sys2 = ss(sys);
A = sys2.A
B = get(sys2,'B')
```

运行结果为①

```
A =
   -0.3333    1.0000   -0.1667
    2.0000         0         0
         0    1.0000         0

B =
     1
     1
     0
```

16.1.3　LTI 模型的组合

串联、并联和反馈是三种最基本的子系统组合方式。MATLAB 提供了实现这些组合方式的函数,如表 16.4 所示。

实际的系统组合会非常复杂,比如多个多输入多输出的系统之间带反馈的相互连接,对于这种复杂的组合方式,MATLAB 提供了 connect 函数通过组合状态矩阵进行处理,但这部分内容远远超出了"信号与系统"课程的要求,故而不再深入介绍。

① 为节省篇幅,将两个矩阵并列打印,和 MATLAB 显示的格式略有差别。

表 16.4 子系统组合方式

名称和使用方法	说 明
sys=series(sys1,sys2),sys=sys1 * sys2	sys1 和 sys2 串联组成的新系统赋值到 sys
sys=parallel(sys1,sys2),sys=sys1+sys2	sys1 和 sys2 并联组成的新系统赋值到 sys
sys=feedback(sys1,sys2)	sys1 前向 sys2 反馈组成的新系统赋值到 sys

MATLAB 知识点(20)——函数和运算符重载

表 16.4 中列出三种 LTI 模型的组合方式，以并联为例，可以调用 parallel 函数，也可以用运算符“+”实现。回忆前面介绍的数值运算和符号运算，不难发现“+”(加法)对多种数据类型都有定义，MATLAB 正是通过重载函数和运算符简化编程和助记命令。在命令窗口中输入“help plus”，将显示

```
+   Plus.
    X + Y adds matrices X and Y. X and Y must have the same
    dimensions unless one is a scalar (a 1-by-1 matrix).
    A scalar can be added to anything.

    C = PLUS(A,B) is called for the syntax 'A + B'when A or B is an
    object.
    Overloaded functions or methods (ones with the same name in other
directories)
      help timeseries/plus.m
      help gf/plus.m
      help zpk/plus.m
      help tf/plus.m
      help ss/plus.m
      help frd/plus.m
      help dynamicsys/plus.m
      help fints/plus.m
      help idmodel/plus.m
      help designdev/plus.m
      help cvdata/plus.m
      help uss/plus.m
```

help umat/plus.m
help ufrd/plus.m
help ndlft/plus.m
help icsignal/plus.m
help atom/plus.m
help sym/plus.m
help laurpoly/plus.m
help laurmat/plus.m

Reference page in Help browser
doc plus

可见加法(plus)函数在多个工具箱中对多种数据类型都有定义，MATLAB命令解释程序会自动根据数据类型从这些工具箱中选择并调用正确的函数。

第二节　反馈系统的基本特性及其应用

主教材从四个方面介绍了反馈对改善系统性能的帮助，下面举例演示用MATLAB实现反馈控制的方法。

16.2.1　改善系统频响特性

例16.3(主教材图11-6修改)　某运算放大器的输入/输出模型可简化为一阶系统

$$A(s)=\frac{A\alpha}{s+\alpha}$$

其中 $A=10^5$，$\alpha=10^3$。这是一个低通滤波器，直流增益为 A，带宽为 α。为扩大其带宽，增加反馈通路 $F(s)=\beta$，其中 $\beta=10^{-4}$。现欲放大一信号 $x(t)=0.4\sin(10^3t)+\sin(10^4t)$，比较原放大器和增加反馈通路之后的放大器的输出有何不同，并解释这种变化。

解　§§§MATLAB文件 ex_16_3.m

```
A = 1e5;
alpha = 1e3;
As = tf(A * alpha,[1,alpha]);          %建立前向通路模型 A(s)
beta = 1e-4;
Fs = tf(beta,1);                       %建立反馈通路模型 F(s)
```

```
Hs = feedback(As,Fs);                          %整个系统模型 H(s)
omg = logspace(0,5,100);                       %欲计算响应的频点
Aomg = freqs(As.num{:},As.den{:},omg);         %计算两个模型的频率响应
Homg = freqs(Hs.num{:},Hs.den{:},omg);
t = 0:1e-5:5e-2;                               %仿真时间抽样点
x = .4 * sin(1e3 * t) + sin(1e4 * t);          %输入信号
y1 = lsim(As,x,t);                             %分别对两个系统进行仿真
y2 = lsim(Hs,x,t);
ex_16_3_plot( );
```

程序运行后，输入信号和两个输出信号的波形如图 16.1 所示。可见加入反馈通路之前，高频分量放大的幅度明显要小于低频分量，因而信号失真严重。加入反馈通路，虽然信号整体放大幅度减小了，但放大时基本无失真。这一点在两个系统的频响曲线上可以更清楚地看到，如图 16.2 所示，加入反馈后虽然低频增益减小，但滤波器工作范围增大了 10 倍，变为 $\alpha(1-\beta A)$，从而能够无失真地放大原信号。

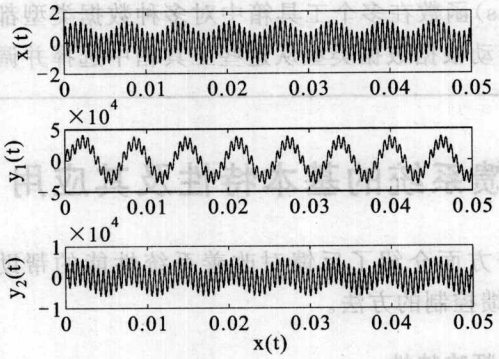


图 16.1 例 16.3 输入输出信号波形

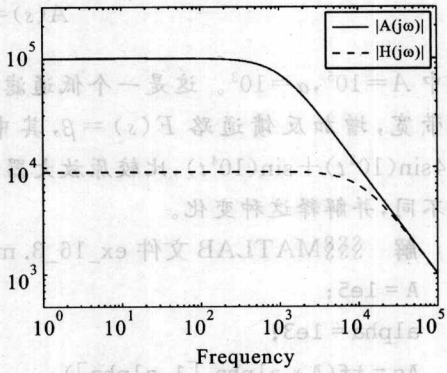


图 16.2 例 16.3 幅频响应曲线

本例也可以用 Simulink 实现，其系统结构如图 16.3 所示。读者可自行搭建，Scope 的输入和图 16.1 完全相同，因而不再重复绘出。

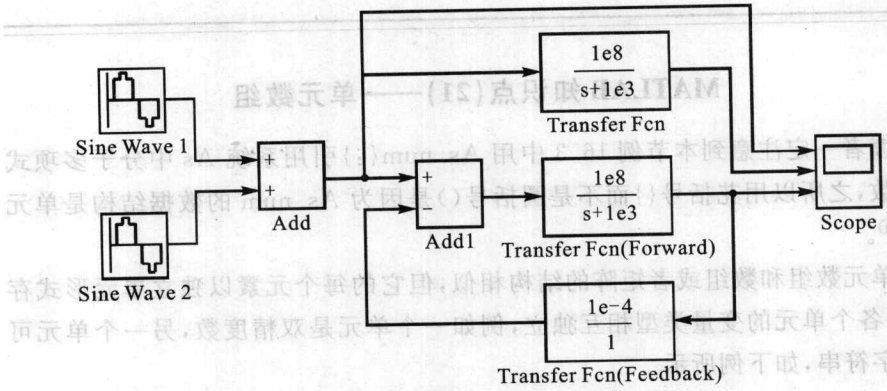


图 16.3　例 16.3 用 Simulink 实现的结构框图

16.2.2　使不稳定系统成为稳定系统

例 16.4(主教材例 11-2 修改)　若系统函数 $A(s)=\dfrac{1}{s-1}$，此系统在右半平面有一阶极点，为不稳定系统，试引入负反馈使其稳定。

解　取反馈 $F(s)=k$，容易写出新的系统函数为

$$H(s)=\frac{1}{s-(1-k)}$$

可见 $k>1$ 时系统稳定，现用 MATLAB 计算验证如下。

§§§MATLAB 文件 ex_16_4.m

```
txt = ['不稳定';'稳定'];
As = tf(1,[1 -1]);                          %用传递函数定义系统 A(s)
As1 = zpk(As);                              %将其转为零点-极点-增益系统
disp(['A(s)系统',txt(1 + (As1.p{:}<0),:)]);
                                            %根据极点位置判断其是否稳定
k = 2;                                      %取 k>1
Fs = tf(k,1);                               %用传递函数定义反馈通道 F(s)
Hs = feedback(As,Fs);                       %构造反馈系统 H(s)
Hs1 = zpk(Hs);                              %转换为零点-极点-增益系统
disp(['H(s)系统',txt(1 + (Hs1.p{:}<0),:)]);
                                            %根据极点位置判断其是否稳定
```

程序运行输出为

```
A(s)系统不稳定
H(s)系统稳定
```

MATLAB 知识点(21)——单元数组

读者一定注意到本节例 16.3 中用 As.num{:}引用系统 As 中分子多项式的系数,之所以用花括号{}而不是圆括号()是因为 As.num 的数据结构是单元数组[①]。

单元数组和数组或者矩阵的结构相似,但它的每个元素以独立单元形式存在,即各个单元的变量类型相互独立,例如一个单元是双精度数,另一个单元可能是字符串,如下例所示

```
A = {'sun',1;[15,2],[1;8]}
```

运行后回显

```
A =
    'sun'              [          1]
    [1x2 double]    [2x1 double]
```

可见不同类型和大小的变量分别占据了 A 的一个单元,而 A 是一个 2×2 的单元数组。

单元数组的定义方式和引用格式也和数组相同,只不过把圆括号()改为花括号{}。例如 A{1,1}表示单元数组 A 的第一行第一列的单元。单元数组一般用于描述句柄或者模型等需要把多个不同类型变量融合起来的环境。可以用 cell 定义单元数组,用 celldisp 和 cellplot 访问和绘制单元数组。但一般而言,用花括号直接定义和访问是使用单元数组的最简捷方式。

第三节 根轨迹

对于图 16.4 所示系统,MATLAB 提供了 rlocus(sys)函数用于绘制闭环系统极点随反馈增益 K 变化时在 s 平面移动的路径,即根轨迹,其中 sys 表示前向通路 $A(s)$的系统模型。另外还有 rlocfind 函数支持用户用鼠标选择并计算感兴趣的极点处的增益。下面两例将说明这两个函数的使用方法。

例 16.5(主教材例 11-7 修改) 对于图 16.5 所示的反馈系统,绘制根轨迹图并计算满足系统稳定的增益 K 的取值范围。

解 此系统的开环系统函数为

① 英文名称 cell array,也有译为"单元阵列"和"元胞"等。

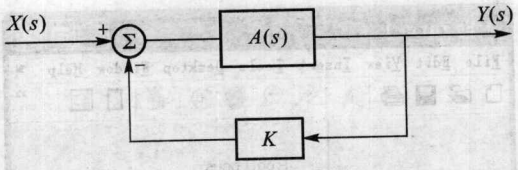


图 16.4　应用 rlocus 函数的系统框图

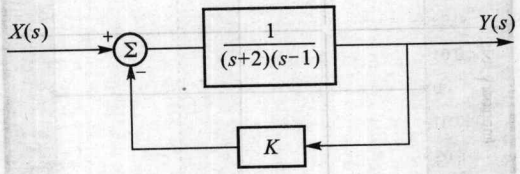


图 16.5　例 16.5 的系统框图

$$A(s)F(s)=\frac{K}{(s+2)(s-1)}$$

首先绘制出根轨迹，然后用鼠标点击选择根轨迹上离虚轴最近的点，从而得到 K 的最大值。在命令窗口输入

```
AF = zpk([],[-2,1],1);    %建立开环系统模型
rlocus(AF);               %绘制根轨迹
K = rlocfind(AF)          %用鼠标在根轨迹上选择关心的极点并计算其
                          %增益
```

程序运行后将显示如图 16.6 所示的窗口，同时命令窗口中将提示

```
Select a point in the graphics window
```

图 16.6 中十字线为鼠标位置，移动鼠标到虚轴和根轨迹的交点，单击左键，即得到图 16.7，其中十字标记为刚才选择的根轨迹上的点，同时命令窗口中将打印出

```
selected_point =
  -0.0138 + 0.0000i
K =
   2.0136
```

所以保证系统稳定的增益范围是 $0<K<2.0136$。受鼠标操作误差的影响，这种方法得到的最大增益并不很准确。下例中将用数值方法计算更加精确的最大增益。

请注意在 MATLAB 中图 16.7 的两条根轨迹显示为不同的彩色线条，很容易区分，但这里打印为黑色线条。为了清楚起见，用箭头标明了三条根轨迹的方

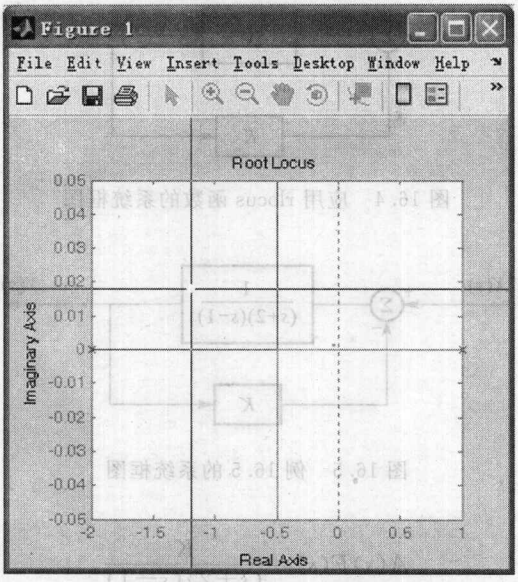


图 16.6 例 16.5 中用鼠标选择根轨迹上的点

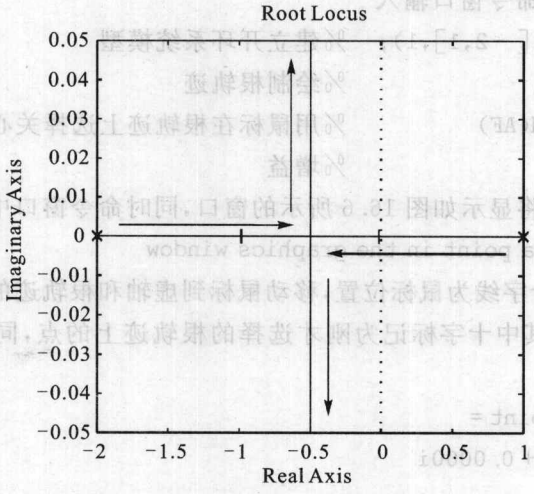


图 16.7 例 16.5 的根轨迹及鼠标选中的点

向，可以看出它们的起点就是开环函数的极点，这和理论结果相同。

例 16.6(主教材例 11-8 修改) 已知反馈系统结构如图 16.8 所示，试绘制其根轨迹图，并计算满足系统稳定的增益 K 的取值范围。

解 此系统的开环系统函数为

$$A(s)F(s)=\frac{K}{s(s+2)(s+4)}$$

下面将使用类似于数值算法中“二分法”的方法寻找所有根轨迹和虚轴的交点。但为加速收敛，一次计算根轨迹上的 16 个值。

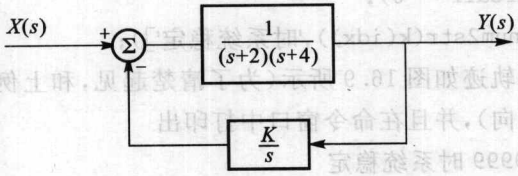


图 16.8　例 16.6 的系统框图

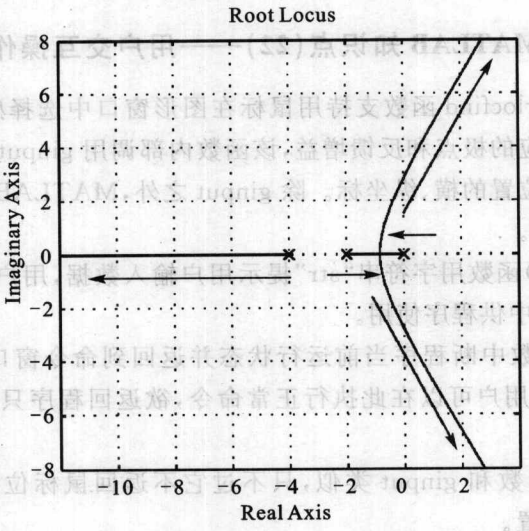


图 16.9　例 16.6 的根轨迹

§§§MATLAB 文件 ex_16_6.m

```
AF = zpk([],[0, - 2, - 4],1);        %建立开环系统模型
rlocus(AF);                          %绘制根轨迹图
[r,k] = rlocus(AF);                  %计算若干个增益 K 对应的极点 r
dk = 1;                              %为满足循环条件
while dk>0.001                       %希望所求 K 和真值的误差小于 0.001
  maxrealr = max(real(r),[],1);      %计算若干条根轨迹中的最大值
  idx = sum(maxrealr< = 0);          %确定左半平面最接近虚轴的点的下标
  dk = [k(idx + 1) - k(idx)]/16;     %将新的 K 值区间 16 等分
```

```
    [r,k] = rlocus(AF,[k(idx):dk:k(idx + 1)]);
                                    %计算对应于新的 K 值的极点 r
end
maxrealr = max(real(r),[],1);  %最后一次确定精确的 K 值下标
idx = sum(maxrealr< = 0);
disp(['0<K<',num2str(k(idx)),'时系统稳定']);
```

系统运行后绘制根轨迹如图 16.9 所示(为了清楚起见,和上例一样用箭头标明了三条根轨迹的方向),并且在命令窗口中打印出

0<K<47.9999 时系统稳定

这和理论值 48 的误差在设定的阈值 0.001 之内。

MATLAB 知识点(22)——用户交互操作

本节介绍的 rlocfind 函数支持用鼠标在图形窗口中选择感兴趣的点,并计算在根轨迹上对应的极点和反馈增益,该函数内部调用 ginput 函数提示十字线并返回鼠标点击位置的横、纵坐标。除 ginput 之外,MATLAB 还提供了多种用户输入/输出函数。

r=input(str)函数用字符串"str"提示用户输入数据,用户输入并回车后数据将被保存在"r"中供程序使用。

keyboard 函数中断程序当前运行状态并返回到命令窗口,同时提示符由">>"变成"K>>",用户可以在此执行正常命令,欲返回程序只要输入 return 函数即可。

gtext(str)函数和 ginput 类似,只不过它不返回鼠标位置而是将字符串"str"显示在该位置。

第四节 小结

本章介绍了用 MATLAB 搭建和研究反馈系统的基本方法。第一节在前文讲解的 tf 和 ss 基础上又补充介绍了另两种描述方法,并就访问 LTI 模型属性和 LTI 模型组合等问题进行了统一说明;第二节研究反馈系统的基本特性及其应用,重点就改善系统频响特性和稳定性两个方面予以举例说明;第三节介绍了根轨迹的绘制方法和交互式计算根轨迹上某点增益的方法。

练习题

1.（主教材习题 11-16）反馈系统的开环函数表达式为

$$A(s)F(s)=\frac{K(s+2)}{s^2+2s+4}\quad(K>0)$$

（1）画出根轨迹；

（2）求两分支的交叉点；

（3）要使闭环系统的冲激响应不呈现振荡，求 K 值范围。

第十七章　系统的状态变量分析

本章介绍用状态空间分析系统的方法。由于第三章已经引入了状态空间的概念以及仿真方法，本章主要就状态变量的线性变换和系统的可控性、可观性分两节进行讨论。

本章包括两个 MATLAB 知识点。① 搜索顺序和搜索路径；② 字符串。

学习重点：系统的可控性、可观性判断方法。

本章正文中首次出现的 MATLAB 函数及其功能

函　数	功　能	函　数	功　能
ss2ss	状态变量的线性变换	canon	对 ss 系统模型进行(如约当化)变换
ctrb	计算可控阵	rank	求矩阵的秩
obsv	计算可观阵		

第一节　状态矢量的线性变换

对状态矢量进行线性变换

$$\boldsymbol{\gamma}=\boldsymbol{P\lambda} \tag{17.1}$$

后，新的系数矩阵 $\hat{\boldsymbol{A}},\hat{\boldsymbol{B}},\hat{\boldsymbol{C}},\hat{\boldsymbol{D}}$ 和原矩阵 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 之间满足

$$\hat{\boldsymbol{A}}=\boldsymbol{PAP}^{-1} \qquad \hat{\boldsymbol{B}}=\boldsymbol{PB} \qquad \hat{\boldsymbol{C}}=\boldsymbol{CP}^{-1} \qquad \hat{\boldsymbol{D}}=\boldsymbol{D} \tag{17.2}$$

上述转换可以通过矩阵运算直接进行，但为简化程序，MATLAB 提供了 sys1＝ss2ss(sys,P)函数描述状态矢量线性变换后系统的改变，其中 sys 为原系统模型，sys1 为新的系统模型。

例 17.1(主教材例 12－17)　给定系统的状态方程为

$$\dot{\boldsymbol{\lambda}}(t)=\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}\boldsymbol{\lambda}(t)+\begin{bmatrix}1\\ 2\end{bmatrix}e(t) \tag{17.3}$$

求在式(17.4)线性变换下的新的状态方程。

$$\begin{cases}\gamma_1=\lambda_1+\lambda_2\\ \gamma_2=\lambda_1-\lambda_2\end{cases} \tag{17.4}$$

解　给定变换矩阵为

$$\boldsymbol{P}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{17.5}$$

因为题目中没有给出输出方程，所以把 $\boldsymbol{C}$,$\boldsymbol{D}$ 矩阵都写成零矩阵。

§§§ MATLAB 文件 ex_17_1.m

```
A=[0,1;-2,-3];
B=[1;2];
C=[0 0];
D=0;
sys=ss(A,B,C,D);
P=[1,1;1,-1];
sys1=ss2ss(sys,P);                    %状态矢量变换
sys1                                  %打印新的状态方程系数矩阵
```

运行程序后输出[①]

```
a =       x1   x2        b =       u1        c =       x1   x2        d =       u1
   x1    -2    0           x1     3              y1     0    0           y1     0
   x2     3   -1           x2    -1
Continuous-time model.
```

所以给定变换下新的状态方程为

$$\dot{\boldsymbol{\gamma}}(t)=\begin{bmatrix}-2 & 0\\ 3 & -1\end{bmatrix}\boldsymbol{\gamma}(t)+\begin{bmatrix}3\\ -1\end{bmatrix}e(t) \tag{17.6}$$

由于 $\boldsymbol{A}$ 矩阵对角化后可以独立研究系统参数对状态变量的影响，所以 MATLAB 提供了 sys1=canon(sys,'model')函数把系统变成规范形式，即 sys1 中的 $\boldsymbol{A}$ 矩阵将约当化。如果调用形式为[sys1,P]=canon(sys,'model')，返回值 P 就是实现 $\boldsymbol{A}$ 约当化的变换矩阵。

例 17.2(主教材例 12-18 修改)　把式(17.7)所示系统的 $\boldsymbol{A}$ 矩阵对角化。

① 为节省篇幅，将四个矩阵并列打印，和 MATLAB 显示的格式略有差别。

$$\dot{\boldsymbol{\lambda}}(t)=\begin{bmatrix}-5 & -1\\ 3 & -1\end{bmatrix}\boldsymbol{\lambda}(t)+\begin{bmatrix}2\\ 5\end{bmatrix}e(t) \tag{17.7}$$

解 §§§MATLAB 文件 ex_17_2.m

```
A=[-5,-1;3,-1];
B=[2;5];
C=[0 0];
D=0;
sys=ss(A,B,C,D);
[sys1,P]=canon(sys,'model');
```

程序运行后根据新的 $\boldsymbol{A}$,$\boldsymbol{B}$ 矩阵可知新的状态方程为

$$\dot{\boldsymbol{\gamma}}(t)=\begin{bmatrix}-4 & 0\\ 0 & -2\end{bmatrix}\boldsymbol{\gamma}(t)+\begin{bmatrix}-7.7782\\ -11.0680\end{bmatrix}e(t) \tag{17.8}$$

和主教材中理论计算的答案相比,$\boldsymbol{A}$ 矩阵对角线上元素的顺序颠倒了,因而 $\boldsymbol{B}$ 矩阵也完全改变了模样。读者可以自己根据 $\boldsymbol{P}$ 矩阵验证这两种结果都是正确的。无法指定 $\boldsymbol{A}$ 矩阵对角化后的状态变量顺序是数值化方法的缺点。但如果可以人工干预的话,仍然可以简单地得到和主教材状态矢量顺序一样的系统,即

```
Q=[0,1;1,0];
sys2=ss2ss(sys1,Q);
```

MATLAB 知识点(23)——搜索顺序和搜索路径

ss 是 MATLAB 用状态方程建立 LTI 模型的函数名,由于其字符较少,很可能在编程时被定义成变量名;另外,也可能有用户自定义的函数文件名为 ss.m。当 MATLAB 命令解释程序遇到字符串 ss 时,它首先在当前工作空间中寻找;如果不是用户定义的变量名,MATLAB 会进一步在内部定义的常量列表中寻找;若还找不到,则认为 ss 是文件名,在当前工作路径下依次寻找 ss.mex 和 ss.m,找到则执行,找不到则依次在所有搜索路径列表下寻找这两个文件。

在命令窗口中输入 path 将列出 MALTAB 的所有搜索路径,MATLAB 通过该列表支持用户调用各个函数组或者工具箱中丰富的函数文件。addpath 和 rmpath 用于在列表中添加或者删除一个路径。为方便灵活地设置搜索路径列表,可以输入 pathtool,或者在主菜单中选择 File→Set Path,将弹出一个对话框显示完整的搜索路径列表,并且支持添加和删除路径,以及调整某条路径在列表中的位置。

如果经过上述全部的搜索过程还没有找到 ss，MATLAB 会在命令窗口中提示输入错误，认为 ss 还没有定义为变量或者函数。为避免混淆，切忌用 MATLAB 常量名或者已有文件名来命名自定义的变量或者函数。

第二节　系统的可控制性与可观测性

系统的可控制性判别矩阵（简称可控阵）

$$\boldsymbol{M}=[\boldsymbol{B}\ \boldsymbol{AB}\ \cdots\ \boldsymbol{A}^{k-1}\boldsymbol{B}] \tag{17.9}$$

满秩，是连续时间系统完全可控的充要条件。MATLAB 提供了 M＝ctrb(sys)函数或者 M＝ctrb(A,B)用于计算可控阵。已知可控阵 M 后，即可用 n＝rank(M)判断其是否满秩。

例 17.3(主教材例 12－23)　给定下列两系统

$$\begin{bmatrix}\dot{\lambda}_1(t)\\ \dot{\lambda}_2(t)\end{bmatrix}=\begin{bmatrix}1 & 1\\ 0 & -1\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}e(t) \tag{17.10}$$

$$\begin{bmatrix}\dot{\lambda}_1(t)\\ \dot{\lambda}_2(t)\end{bmatrix}=\begin{bmatrix}1 & 1\\ 2 & -1\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}e(t) \tag{17.11}$$

问这两系统是否都可控？

解　§§§ MATLAB 文件 ex_17_3.m

```
txt = ['不完全可控';'完全可控  '];          %注意“完全可控”四字后有一
                                            %个空格
A1 = [1,1;0,-1];
B1 = [1;0];
M1 = ctrb(A1,B1);                           %生成可控阵
n1 = rank(M1);                              %计算可控阵的秩
disp(['系统 1',txt(1+(n1==2),:)]);          %不满秩显示前一个字串，否
                                            %则显示后一个
A2 = [1,1;2,-1];
B2 = [0;1];
M2 = ctrb(A2,B2);
n2 = rank(M2);
disp(['系统 2',txt(1+(n2==2),:)]);
```

运行程序后输出

系统 1 不完全可控

系统 2 完全可控

系统的可观测性判别矩阵(简称可观阵)

$$\boldsymbol{N}=\begin{bmatrix}\boldsymbol{C}\\ \boldsymbol{CA}\\ \vdots\\ \boldsymbol{CA}^{k-1}\end{bmatrix} \tag{17.12}$$

满秩,是连续时间系统完全可观的充要条件。同样,MATLAB 提供了 N=obsv(sys)函数或者 N=obsv(A,C)用于计算可观阵。

例 17.4(主教材例 12-25) 讨论给定系统的可观性。

$$\begin{cases}\begin{bmatrix}\dot{\lambda}_1(t)\\ \dot{\lambda}_2(t)\end{bmatrix}=\begin{bmatrix}1 & 1\\ -2 & -1\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}e(t)\\ r(t)=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}\end{cases} \tag{17.13}$$

解 §§§ MATLAB 文件 ex_17_4.m

```
txt = ['不完全可观';'完全可观  '];          %注意"完全可观"四字后有一
                                            %个空格
A = [1,1; -2, -1];
C = [1,0];
N = obsv(A,C);                              %生成可观阵
n = rank(N);                                %计算可观阵的秩
disp(['系统 1 ',txt(1 + (n == 2),:)]);
```

运行程序后输出

系统 1 完全可观

MATLAB 知识点(24)——字符串

字符串是 MATLAB 的基本数据类型之一,用单引号' '限定,例如在命令窗口中输入

```
s = 'abc012 好'
```

将把 s 定义为一个字符串变量。和 C 语言不同,MATLAB 中每个字符都用

UNICODE 码表示，例如输入

```
a = double(s)
```

将回显

```
a =
97   98   99   48   49   50   22909
```

可见 a 的每个值对应于 s 中该字符的 UNICODE 码。上例暗示着字符串以行矢量形式存储，如果想把两个字符串拼接起来，既可以用函数 strcat，也可以用矩阵定义的方法，例如

```
s1 = [s,'tail']
```

则 sl 变成“abc012 好 tail”，同理可以把字符串写在几行，但要求每行字符个数相同[①]，这一点已经在例 17.3 和例 17.4 中看到。可以用 setstr(a)函数将数值转换回字符串，也可以用和标准 C 函数重名的 sprintf，若要把字符串转换为数值，则用 str2num，类似的还有很多字符串函数，请用 help strfun 仔细学习。

第三节　小结

本章第一节分别举例介绍了状态变量的线性变换方法和状态变量的对角化；第二节分别讨论了系统的可控性和可观性的判决方法，MATLAB 提供了 ctrb 和 obsv 函数计算系统的可控阵和可观阵，极大地简化了可控性和可观性的判别。

练习题

1.（主教材习题 12-21 修改）给定线性时不变系统的状态方程和输出方程

$$\begin{cases}\dot{\boldsymbol{\lambda}}(t)=\begin{bmatrix}-2 & 2 & -1\\ 0 & -2 & 0\\ 1 & -4 & 0\end{bmatrix}\boldsymbol{\lambda}(t)+\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}e(t)\\ r(t)=\begin{bmatrix}1 & 0 & 0\end{bmatrix}\boldsymbol{\lambda}(t)\end{cases}$$

试判断该系统的可控性和可观性，并求出系统函数。

① 因为使用 UNICODE，汉字也表示为单字符。

第十八章　控制系统仿真

本章将分别研究潜水艇下潜和倒立摆平衡两个实际的控制系统。和本书前面三次大作业稍有不同，本次仿真实验将给出比较详细的解答以保证读者可以顺利地完成实验并掌握相关的背景知识[①]。

第一节　潜水艇下潜控制

潜水艇的控制系统如图 18.1 所示，实际深度 $c(t)$ 可以用压力传感器测出，并和期望深度 $r(t)$ 进行比较，两者之间的差异被用来控制尾翼调节器，调整尾翼角度进而导致上浮或下潜。

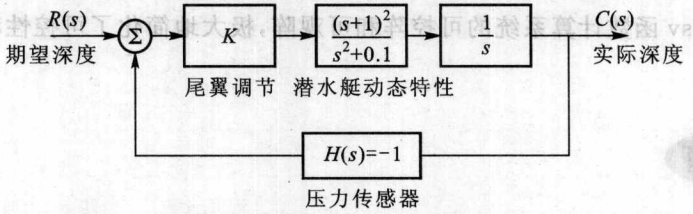


图 18.1　潜水艇的系统框图

(1) 以 $r(t)$ 为输入，$c(t)$ 为响应，求系统的传递函数。

根据系统框图可直接写出传递函数，化简后为

$$\frac{C(s)}{R(s)}=\frac{Ks^2+2Ks+K}{s^3+Ks^2+(2K+0.1)s+K} \tag{18.1}$$

(2) 写出系统的状态方程和输出方程。

主教材第十二章 12.2 节介绍了一种先根据传递函数绘出流图，再选择积分

① 目前，国内较多院校在本课程中留给反馈和状态变量的学时较少，考虑这一实际情况，本章的解题分析更详尽，以减少自学之困难。

器的输出作为状态变量，从而写出状态方程和输出方程的方法。下面介绍通过中间变量将传递函数直接改写为状态方程的方法，首先定义 $\Lambda(s)$

$$\Lambda(s)=\frac{C(s)}{Ks^2+2Ks+K}=\frac{R(s)}{s^3+Ks^2+(2K+0.1)s+K} \tag{18.2}$$

根据上式可以写出关于输入和输出的两个时域表达式

$$c(t)=K\frac{\mathrm{d}^2\lambda(t)}{\mathrm{d}t^2}+2K\frac{\mathrm{d}\lambda(t)}{\mathrm{d}t}+K\lambda(t) \tag{18.3}$$

$$r(t)=\frac{\mathrm{d}^3\lambda(t)}{\mathrm{d}t^3}+K\frac{\mathrm{d}^2\lambda(t)}{\mathrm{d}t^2}+(2K+0.1)\frac{\mathrm{d}\lambda(t)}{\mathrm{d}t}+K\lambda(t) \tag{18.4}$$

定义三个状态变量 $\lambda_1(t)=\lambda(t)$，$\lambda_2(t)=\dfrac{\mathrm{d}\lambda(t)}{\mathrm{d}t}$，$\lambda_3(t)=\dfrac{\mathrm{d}^2\lambda(t)}{\mathrm{d}t^2}$，从而可以写出状态方程

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\\ \lambda_3(t)\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -K & -(2K+0.1) & -K\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\\ \lambda_3(t)\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}r(t) \tag{18.5}$$

和输出方程

$$c(t)=\begin{bmatrix}K & 2K & K\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\\ \lambda_3(t)\end{bmatrix}+[0]r(t) \tag{18.6}$$

MATLAB 的 ss 函数除了通过状态方程定义 LTI 系统外，还可以把其他方式描述的 LTI 系统转为由状态方程描述，对本题而言，在命令窗口输入（因为 LTI 模型中不支持用变量作为系数，假设 $K=1$）

```
sys = tf([1,2,1],[1,1,2.1,1]);
ss(sys)
```

将看到 MATLAB 回显[①]。

```
a =
      x1     x2     x3     u1
x1   -1  -1.05   -0.5     x1  2
x2    2      0      0     x2  0
x3    0      1      0     x3  0

b =
      u1
x1     2
x2     0
x3     0

c =
      x1   x2   x3
y1   0.5  0.5 0.25

d =
      u1
y1     0

Continuous-time model.
```

① 为节约篇幅，将四个矩阵并列打印，和 MATLAB 显示的格式略有差别。

即对应于状态方程①。

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\gamma_1(t)\\ \gamma_2(t)\\ \gamma_3(t)\end{bmatrix}=\begin{bmatrix}-K & -(K+0.05) & -0.5K\\ 2 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\gamma_1(t)\\ \gamma_2(t)\\ \gamma_3(t)\end{bmatrix}+\begin{bmatrix}2\\ 0\\ 0\end{bmatrix}r(t) \tag{18.7}$$

和输出方程

$$c(t)=[0.5K \quad 0.5K \quad 0.25K]\begin{bmatrix}\gamma_1(t)\\ \gamma_2(t)\\ \gamma_3(t)\end{bmatrix}+[0]r(t) \tag{18.8}$$

可见其状态和定义的不同，$\gamma_1(t)=2\lambda_3(t)$，$\gamma_2(t)=4\lambda_2(t)$，$\gamma_3(t)=4\lambda_1(t)$。

(3) 假设潜水艇要潜入 60 m 深，请针对 $K=0.45$、$K=2$ 和 $K=10$ 三种情况，仿真其潜水过程，并绘制 $c(t)$变化曲线。这三种情况的系统极点分别是多少？能否解释你看到的下潜曲线？

用 Simulink 搭建的仿真模型如图 18.2 所示，可见它和系统框图非常相似，再一次见证了 MATLAB 仿真工具的友善界面。针对 $K=0.45$、2 和 10 三种情况，分别对图 18.2 所示系统进行仿真，结果在图 18.3、图 18.4 和图 18.5 中给出。可见当 $K=0.45$ 时，潜水艇以水下 60 m 为中心做等幅振荡，永远不会停止在 60 m 深度。可以用 MATLAB 求出此时系统的极点

```
sys = tf([0.45,0.9,0.45],[1,0.45,1,0.45]);
ps = pole(sys).'
```

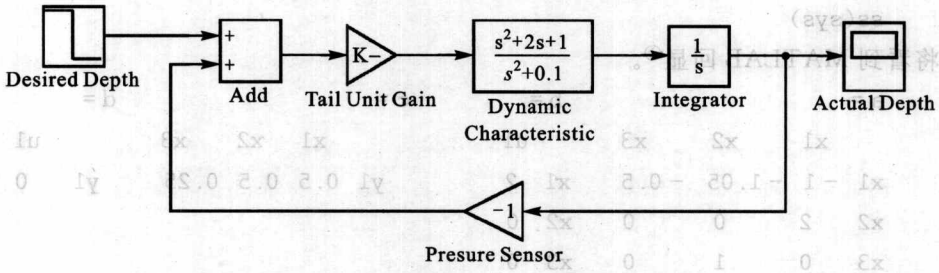


图 18.2 潜水艇的 Simulink 模型图

① 这里要求从常量矩阵中判断出变量 K 的位置，似乎有些困难，但读者只要将 K 代入几个不同的值就不难发现其规律。

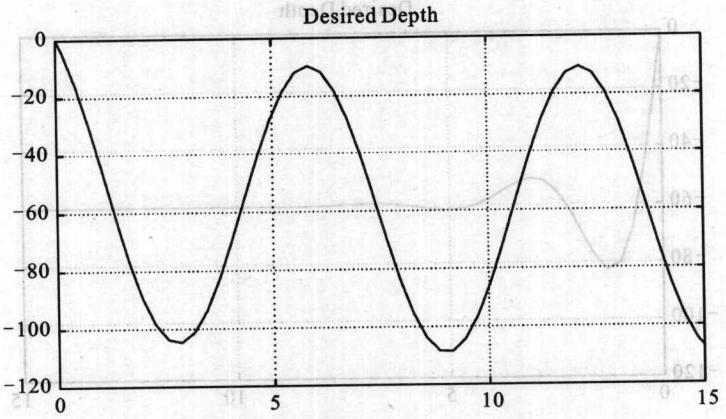


图 18.3　$K=0.45$ 时的下潜曲线

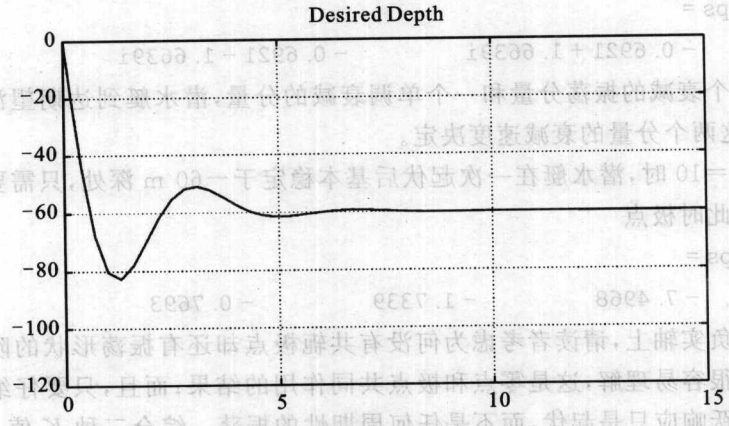


图 18.4　$K=2$ 时的下潜曲线

MATLAB 回显[①]。

```
ps =
   0.0000 + 1.0000i   0.0000 - 1.0000i   -0.4500
```

可见系统在虚轴上有两个极点，因而稳定于振荡状态，而且可知振荡周期为 2π；系统在实轴 -0.45 处也有一个极点，它对应于振荡中心从初始 0 m 到 -60 m 的瞬态。

当 $K=2$ 时，潜水艇经过几次振荡，大约 6 个单位时间后收敛到 -60 m。此时系统极点为

① 为节约篇幅，将极点以行矢量形式显示。

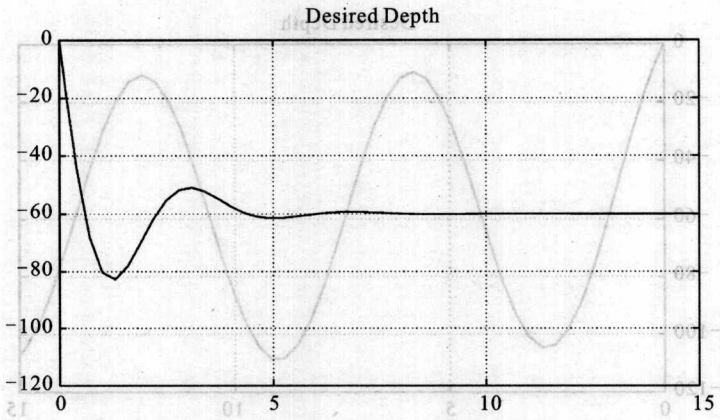


图 18.5 $K=10$ 时的下潜曲线

```
ps =
  -0.6921 + 1.6639i        -0.6921 - 1.6639i        -0.6158
```

即包括一个衰减的振荡分量和一个单调衰减的分量，潜水艇到达期望深度的时间，即由这两个分量的衰减速度决定。

当 $K=10$ 时，潜水艇在一次起伏后基本稳定于 -60 m 深处，只需要两个时间单位。此时极点

```
ps =
  -7.4968          -1.7339          -0.7693
```

全部位于负实轴上，请读者考虑为何没有共轭极点却还有振荡形状的阶跃响应呢？其实很容易理解，这是零点和极点共同作用的结果；而且，只要仔细观察就会发现阶跃响应只是起伏，而不是任何周期性的振荡。综合三种 K 值，可见 10 是保证潜水艇机动性的最佳选择，但实际上较大的 K 值意味着要求更坚固的尾翼以承受巨大的加速度，所以工程中要折中考虑很多因素以确定最佳的系统参数。

本例是一个阶跃响应问题，在主教材 5.4 节有详细讨论，读者不妨对不同 K 值分别求系统的频率特性和阶跃响应，并观察上升时间与系统带宽和超调(即峰超)间的关系。

(4) 请用根轨迹的方法计算保证系统稳定的 K 值范围。

为了使用根轨迹法，将变量 K 移到反馈支路，修正后的系统框图如图 18.6 所示。

在 MATLAB 命令窗口中输入

```
Fs = tf([1 2 1],[1 0 0.1 0]);
rlocus(Fs)
```

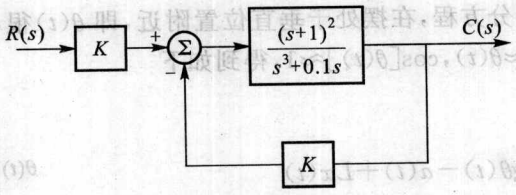


图 18.6　修正后的潜水艇系统框图

可见潜水艇系统有三条根轨迹，一条起于原点处的极点，终止于负实轴上的零点；另外两条分别起于虚轴上的共轭极点，终止于上述零点和虚轴上的负无穷点。再次输入

```
K = rlocfind(Fs)
```

并用鼠标选中根轨迹和虚轴的交点，MATLAB 将回显

```
selected_point =
    0.0013 + 0.9981i
K =
    0.4475
```

考虑到鼠标操作的误差，满足稳定的 K 值应当大于 0.45，验证了前述结果。最终的根轨迹如图 18.7 所示。

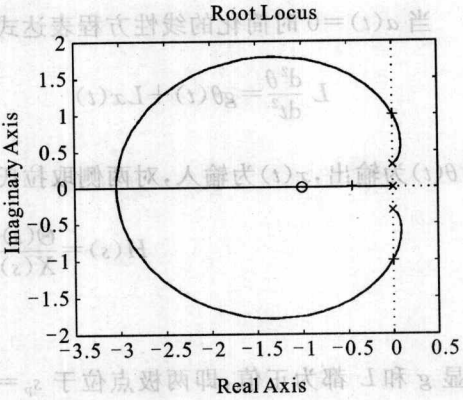


图 18.7　潜水艇系统的根轨迹

第二节　倒立摆平衡控制

请参看主教材第十一章的习题 11-11 的详细提示以及《教与写的记忆——信号与系统评注》(郑君里著)第三篇 3.11-4 节倒立摆稳定性分析。

如图 18.8 所示倒立摆系统，摆长为 L，不计长杆质量，末端小球质量为 m，$\theta(t)$是偏离垂线的角度，重力加速度为 g，$a(t)$是小车的加速度，$x(t)$表示扰动(如风吹)引起的角加速度。质量沿垂直于杆方向的加速度 $L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$应等于沿此方向施加的各种加速度之和，包括重力加速度、小车加速度和扰动加速度，按此要求建立的系统动态方程如下

$$L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}=g\sin\theta(t)-a(t)\cos\theta(t)+Lx(t)$$

此模型为非线性微分方程，在摆处于垂直位置附近，即 $\theta(t)$ 很小的情况下，取如下近似：$\sin[\theta(t)]\approx\theta(t)$，$\cos[\theta(t)]\approx1$，得到如下简化的线性方程

$$L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}=g\theta(t)-a(t)+Lx(t)$$

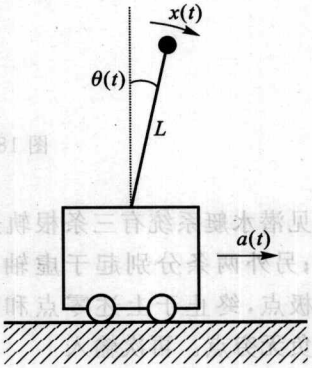


图 18.8 倒立摆系统

(1) 设 $x(t)$ 为激励信号，$\theta(t)$ 是响应信号，若小车不动，即 $a(t)=0$，写成系统函数 $H(s)=\frac{\Theta(s)}{X(s)}$ 表达式，并讨论系统的稳定性。

当 $a(t)=0$ 时简化的线性方程表达式为

$$L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}=g\theta(t)+Lx(t)$$

以 $\theta(t)$ 为输出，$x(t)$ 为输入，对两侧取拉氏变换得到

$$H(s)=\frac{\Theta(s)}{X(s)}=\frac{1}{s^2-\frac{g}{L}}$$

明显 g 和 L 都为正值，即两极点位于 $s_p=\pm\sqrt{\frac{g}{L}}$，其中之一位于右半平面，因而系统不稳定。此时系统框图如图 18.9 所示。

(2) 研究适当移动小车对稳定性的影响。假定随 $\theta(t)$ 的变化按比例反馈作用使小车产生加速度，即 $a(t)=K\theta(t)$，K 为比例系数。画出引入反馈后的系统方框图，并求出此反馈系统的系统函数。讨论系统的稳定性(分为 $K<g$、$K=g$ 和 $K>g$ 三种情况)。

引入反馈使小车移动，令 $a(t)=K\theta(t)$，此时微分方程表达式为

$$L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}=g\theta(t)-K\theta(t)+Lx(t)$$

取拉氏变换后得到

$$H(s)=\frac{\Theta(s)}{X(s)}=\frac{1}{s^2-\frac{g}{L}+\frac{K}{L}}$$

引入比例反馈后系统框图如图 18.10 所示。

此时系统的两个极点位于 $s_p=\pm\sqrt{\frac{g-K}{L}}$。当 $K<g$ 时有一极点位于右半平

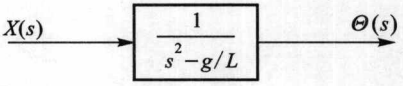


图 18.9　小车静止时倒立摆系统框图

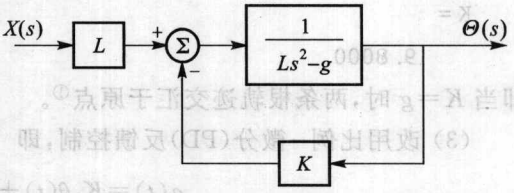


图 18.10　比例反馈时倒立摆系统框图

面，系统不稳定；当 $K=g$ 时在 $s=0$ 处有二阶极点，系统不稳定；当 $K>g$ 时在 $j\omega$ 轴上有共轭极点，系统处于临界稳定状态，倒立摆以无阻尼方式来回摆动。

可以用根轨迹考察 K 取值对系统稳定性的影响，在命令窗口中输入

```
g = 9.8;
L = 1;
Fs = tf(1,[L,0,-g]);
rlocus(Fs);
```

根轨迹如图 18.11 所示，可见有一条根轨迹从极点 3.13 沿着实轴走到原点，然后再沿着虚轴走向正无穷，因而无论 K 取何值，总有一个极点在右半平面或者有两个极点在虚轴上，因而系统要么不稳定，要么临界稳定。再输入

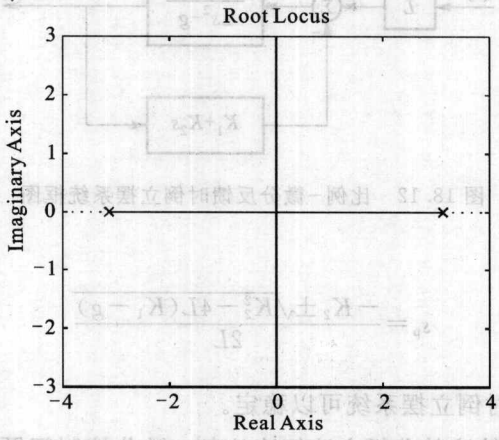


图 18.11　比例反馈时倒立摆系统的根轨迹

```
K = rlocfind(Fs)
```

然后用鼠标点击原点，MATLAB 将输出

```
selected_point =
```

```
0.0095 - 0.0093i

K =

9.8000
```

即当 $K=g$ 时，两条根轨迹交汇于原点[①]。

(3) 改用比例－微分(PD)反馈控制，即

$$a(t)=K_1\theta(t)+K_2\frac{\mathrm{d}\theta}{\mathrm{d}t}$$

其中 K_1 和 K_2 都为正实系数。写出此反馈系统的系统函数，讨论为使系统稳定，K_1，K_2 应满足何种约束条件？

采用比例－微分反馈控制后，系统微分方程表达式为

$$L\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}=g\theta(t)-K_1\theta(t)-K_2\frac{\mathrm{d}\theta(t)}{\mathrm{d}t}+Lx(t)$$

取拉氏变换后求得

$$H(s)=\frac{\Theta(s)}{X(s)}=\frac{1}{s^2+\dfrac{K_2}{L}s+\dfrac{K_1-g}{L}}$$

引入比例－微分反馈后系统框图如图 18.12 所示。

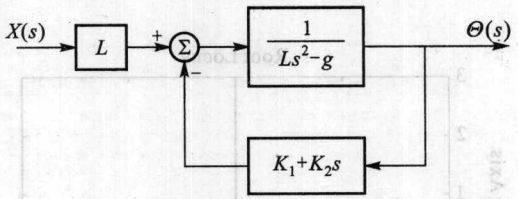


图 18.12 比例－微分反馈时倒立摆系统框图

两个极点位于

$$s_p=\frac{-K_2\pm\sqrt{K_2^2-4L(K_1-g)}}{2L}$$

当 $K_2>0$ 和 $K_1>g$ 时倒立摆系统可以稳定。

为了可以用根轨迹的分析方法解决比例－微分控制问题，约束 $K_1=K_2$，从而系统框图变形为图 18.13。由于末端的子系统极点 $s_p=-1$，因而是稳定的，只需要研究反馈部分即可，即

① selected point 之所以不为零是因为鼠标点击时的误差导致的，但可以看出它的实部和虚部都很小。

```
g = 9.8;
L = 1;
Fs = tf([1 1],[L,0,-g]);
rlocus(Fs);
```

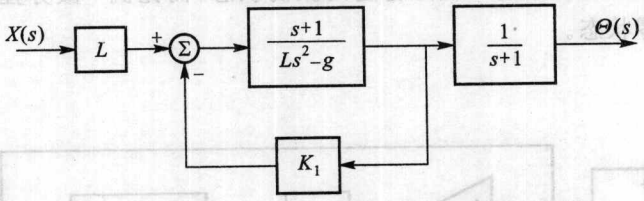


图 18.13　简化后的倒立摆系统框图

根轨迹如图 18.14 所示，有一条根轨迹从极点 3.13 沿着实轴走到 −1 位置的零点，因而 K_1 必须要大于一个门限才能保证两个极点都在左半平面，再输入

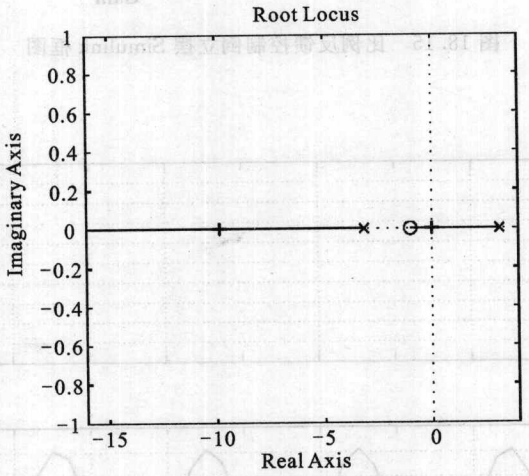


图 18.14　比例−微分反馈时倒立摆系统的根轨迹

```
K1 = rlocfind(Fs)
```

然后用鼠标点击原点，MATLAB 将输出

```
selected_point =
    -0.0047 - 0.0031i
K1 =
    9.8466
```

即当 $K_1>g$ 时，两个极点都在左半平面。

用 Simulink 分别仿真比例反馈控制和简化的比例-微分反馈控制倒立摆系统在扰动条件下的表现，系统框图分别如图 18.15 和图 18.17 所示，选择 $K=K_1=20$，对于输入的扰动信号，系统输出分别如图 18.16 和图 18.18 所示，可见单纯的比例控制让系统进入临界稳定的振荡状态，而比例-微分控制则可以恢复到稳定的零状态。

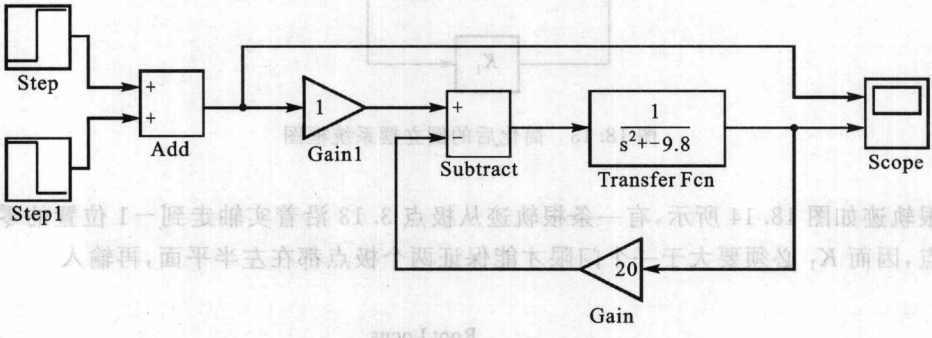


图 18.15 比例反馈控制倒立摆 Simulink 框图

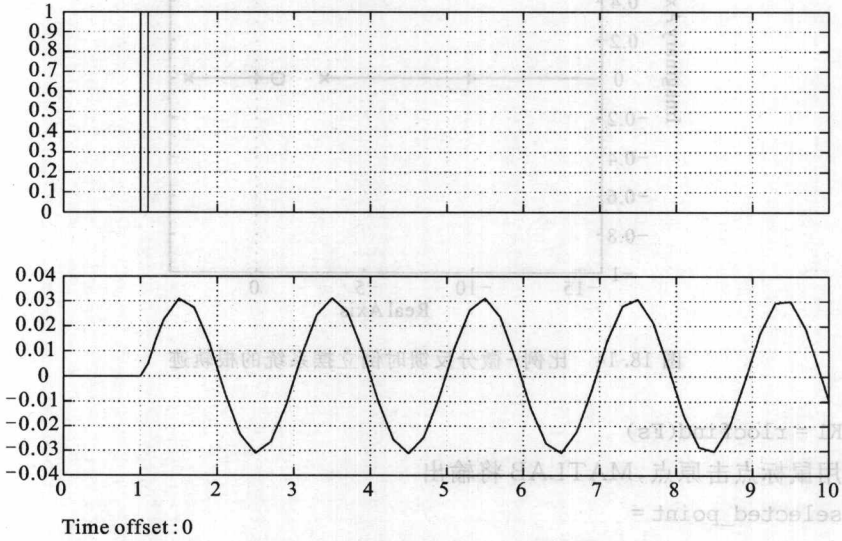


图 18.16 比例反馈控制倒立摆仿真波形

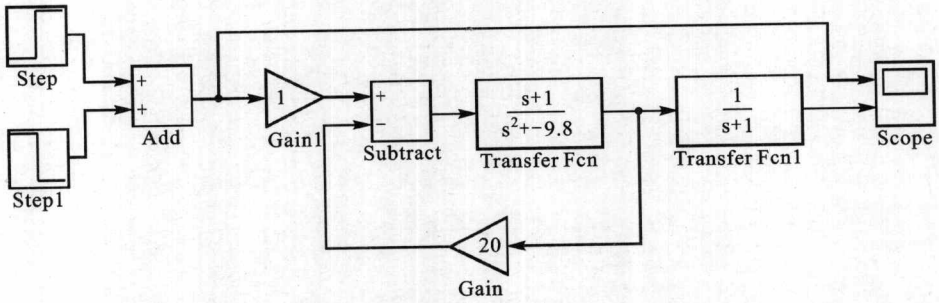


图 18.17　比例-微分反馈控制倒立摆 Simulink 框图

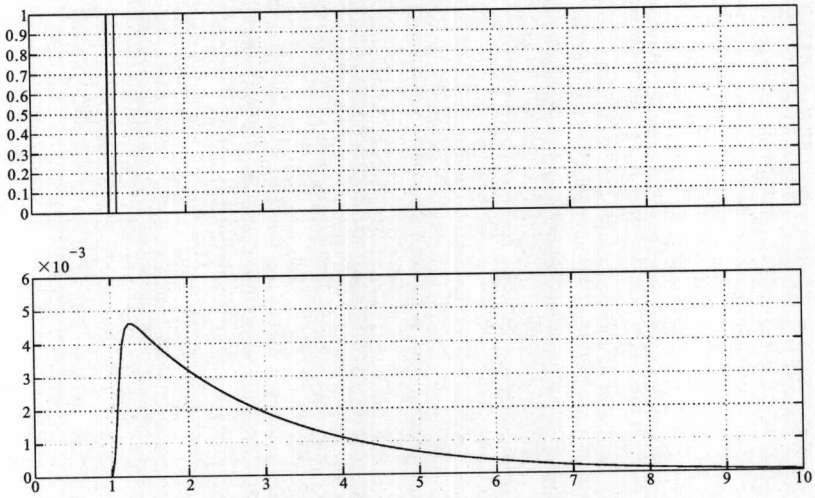
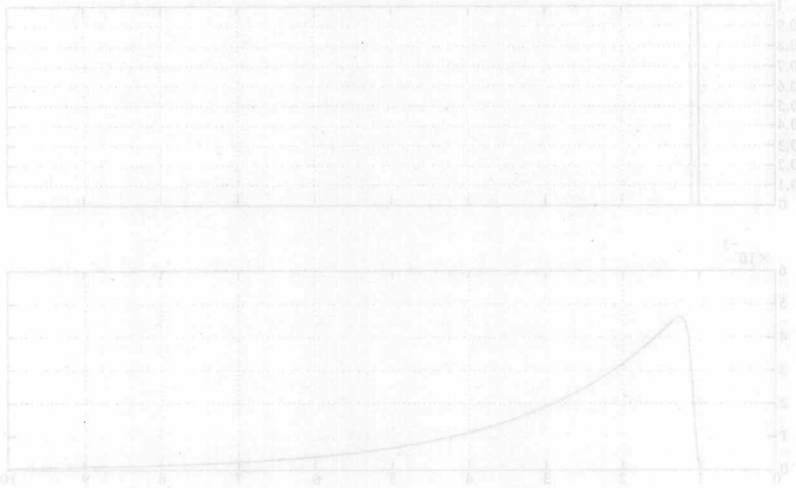


图 18.18　比例-微分反馈控制倒立摆仿真波形

Step
Add
Gain1
Subtract
Transfer Fcn
Transfer Fcn1
Scope
Step1
Gain



第七篇

MATLAB 编程辅导三

第七章

MATLAB 编程指令三

第十九章　图形用户界面(GUI)设计

图形用户界面(Graphical User Interface,GUI)是指由按钮、列表框、编辑框等用户界面控件构成的应用程序界面。在GUI下,用户无需记忆繁琐的命令,只需要通过鼠标、键盘等即可与系统进行直观、便捷的交互。MATLAB提供了功能强大的集成GUI开发环境(Graphical User Interface Development Environment,GUIDE)。利用GUIDE用户可以方便地建立和设计GUI,并通过编程控制GUI和用户之间的交互操作。

GUI是功能程序的包装。为了演示GUI的设计方法,我们必须首先选择一个功能程序。纵观MATLAB提供的各种工具箱,尤其是信号处理工具箱中sptool等各种强大的GUI信号处理工具,不难发觉尚缺少一个查看信号频谱(傅里叶变换)的GUI工具(spectrogram函数可以对信号做短时傅里叶变换,但界面不够友善)。本章将逐步设计一个可以同时绘制工作空间中的信号时域波形和频谱的自定义工具,命名为tfviewer。

第一节　启动GUI

在MATLAB命令窗口中输入"guide",或者在MATLAB工具条中点击guide按钮,都可以启动GUIDE设计向导对话框,如图19.1所示,其中提供了四种GUIDE模板。一般选择缺省的空白GUI即可,确定后弹出如图19.2所示的控件布局编辑器Layout Editor。控件布局编辑器是GUIDE实用工具的集成控制台,主要由上方的工具栏、左下侧的控件面板和右下方大面积的设计区组成。其中控件面板包含了所有的GUI控件,用户可以将其拖拽到右侧设计区,也可以选中某个控件后,再到设计区绘制控件。

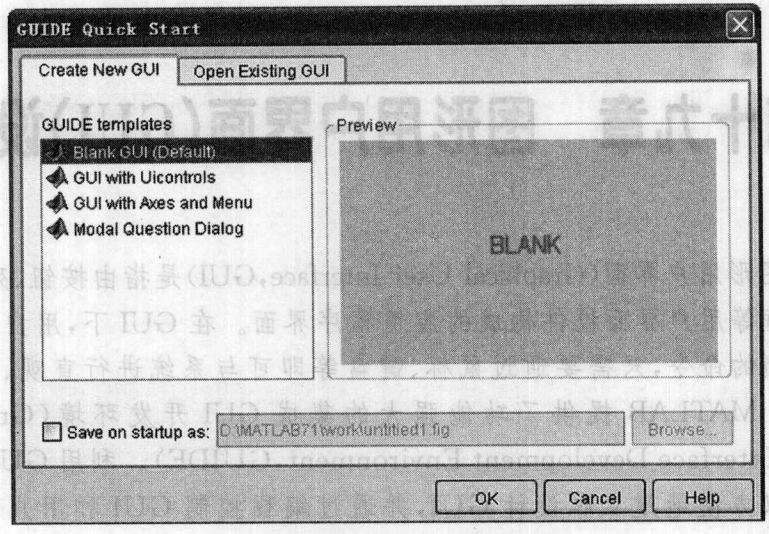


图 19.1 GUIDE 设计向导

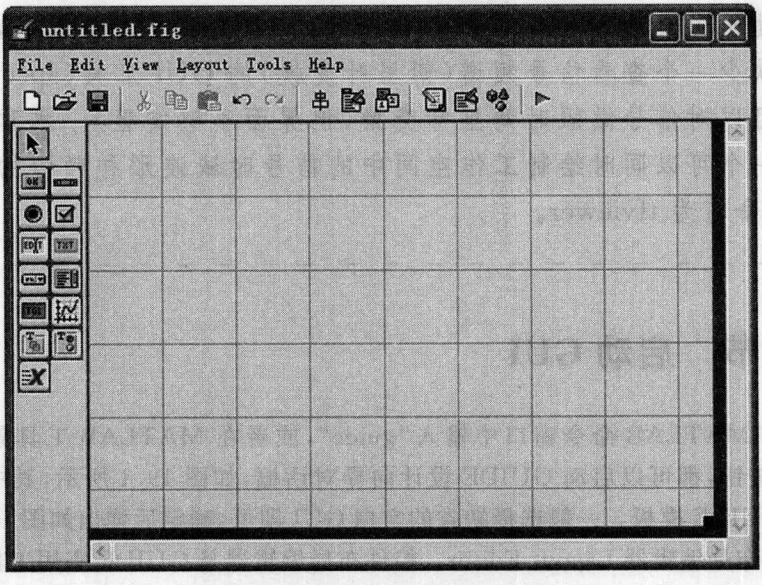


图 19.2 GUIDE 控件布局编辑器

第二节　设计和保存 GUI

为了实现要求的绘制工作空间中信号时域波形和频谱的目的，需要和用户交互确定如下信息：需要绘制哪个变量的波形和频谱？它的采用时间用哪个变量表示？希望观测的频谱范围是多少？该范围内有多少的频率抽样点？为了进一步提高对用户的友善性，还可以主动列出工作空间中的所有变量供用户选择。

根据以上分析，计划用到的 GUI 控件包括：一个列表框用来展现工作空间中的所有变量；两个按钮分别用来指定抽样时间和抽样点对应的变量；两个编辑框分别用来输入频域观测范围和频域抽样点数；两个坐标轴分别用来绘制波形和频谱；最后，还需要两个按钮，一个让用户发出绘图的命令，另一个用来更新工作空间中的变量以方便用户交互式使用。

逐一选择所需控件，并摆放到希望的位置(虽然知道应当摆放美观，但就如何才能摆放美观却无法提出任何有价值的建议，但可以肯定的是，相关控件必须对齐，同类控件大小也要相同)。最终布局结果如图 19.3 所示。现在点击工具条上的“保存”按钮，弹出一个“另存为”的对话框，请输入“tfviewer”，点击“确定”，将布局结果保存为“tfviewer”。

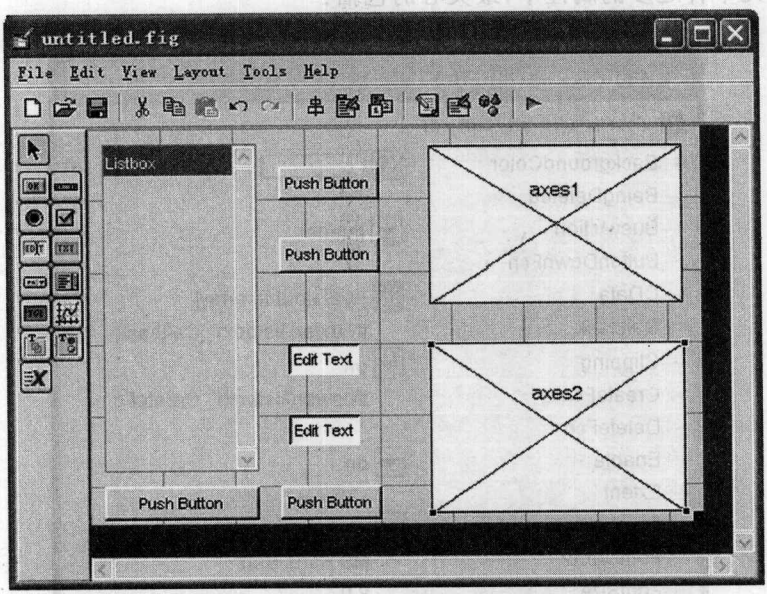


图 19.3　tfviewer 所需控件的初步布局

GUI 被自动保存在扩展名分别为“.fig”和“.m”的两个文件中，其中 FIG 文件用于描述各个控件的类型、属性和布局等信息，M 文件用于保存运行该 GUI 所需的所有程序代码，包括回调函数框架等。这些内容接下来会详细讲解。

第三节 运行 GUI

虽然现在还没有写任何功能程序，但就 GUI 本身而言，它现在就可以运行了。点击控件布局编辑器的工具条上的 run 按钮，会看到屏幕上出现一个和控件布局编辑器一模一样的对话框，可以点击按钮，也可以在编辑框中输入文字，虽然这么做并没有什么意义。

运行这个 GUI 还有两种方法，一是在命令窗口中输入“tfviewer”，二是用编辑/调试器打开并运行刚才生成的 tfviewer.m 文件。

第四节 修改 GUI 控件属性

双击设计区中的空白区域或者任何一个控件（比如列表框），会弹出图 19.4 所示的属性浏览器，每种控件的属性数量和类别各不相同，但有些属性是共有的。在几十种之多的属性中，最关心的包括：

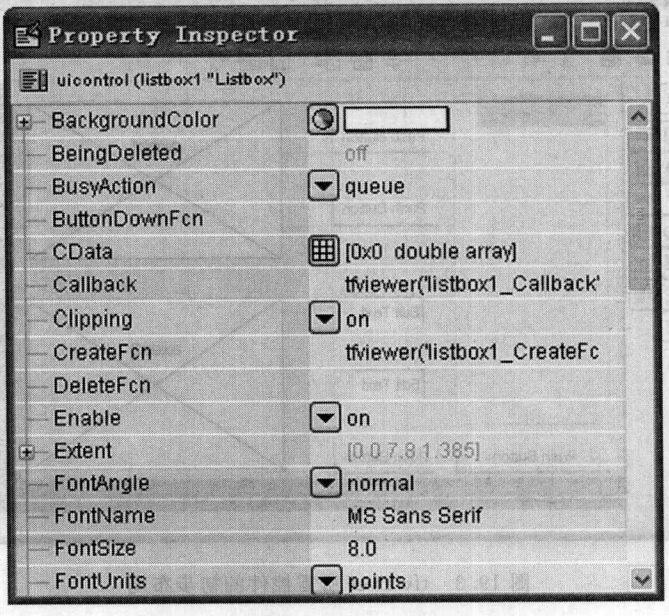


图 19.4 控件属性浏览器

Tag:是该控件的唯一标识,编程控制和访问该控件的唯一途径。

String:按钮上显示的文字,或者编辑框内显示的文字。

Value:列表框内被选中的项的索引。

逐一修改各个控件的相关属性,修改内容如表 19.1 所示。另外,还添加了若干指示功能的静态文本,这里不再赘述。最终的布局如图 19.5 所示。

表 19.1　各个控件的功能和主要属性

类　型	Tag	String	功　能
列表框	lst_Variables	空	列出工作空间中的所有变量
按钮	bt_Refresh	Refresh	更新列表
按钮	bt_SetSampleTime	Set Sample Time	指定表示抽样时间的变量
按钮	bt_SetSampleData	Set Sample Data	指定表示抽样数据的变量
编辑框	ed_FreqRange	空	指定绘制的频率范围
编辑框	ed_FreqSampleNum	空	指定频域绘制的抽样点数
按钮	bt_Draw	Draw	绘制波形和频谱
坐标轴	ax_Time		绘制波形的坐标轴
坐标轴	ax_Freq		绘制频谱的坐标轴

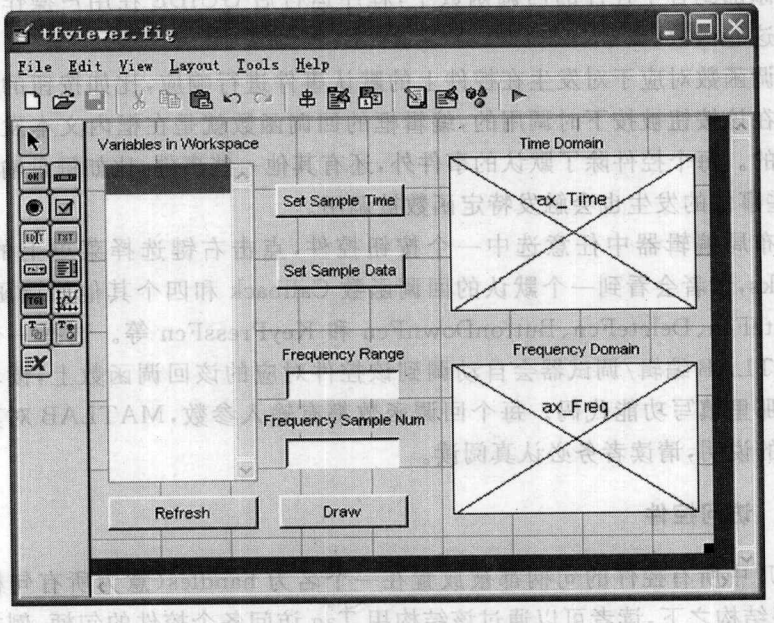


图 19.5　tfviewer 最终控件布局

第五节 编程控制GUI的方法

现在已经完成了GUI的全部设计工作，要开始修改M文件以实现绘制工作空间中变量的波形和频谱的功能。首先介绍M文件的各个组成部分。

19.5.1 OpeningFcn函数

每次GUI启动后，程序会依次初始化各个控件，然后调用OpeningFcn函数(在例子中命名为tfviewer_OpeningFcn)。读者可以在这里添加一些初始化时执行的任务。

和OpeningFcn函数对应的是OutputFcn函数，它是在程序即将退出时被调用的，用于处理返回值和程序的其他收尾工作，本例中不使用，因而不再分项介绍。

19.5.2 回调函数(Callback Function)

用户对GUI控件进行某种操作后，需要M文件执行一段代码以完成该操作对应的功能，这个过程就是通过回调函数实现的。读者只要把需要完成的功能代码添加到各个控件的回调函数中，程序运行后GUIDE在用户操作时会自动调用这些函数。

回调函数对应于对发生在控件上的默认事件进行响应，比如按钮的回调函数就是在该按钮被按下时调用的，编辑框的回调函数就是在框内文本发生变化后调用的。每个控件除了默认的事件外，还有其他一些事件，比如键盘输入事件等，这些事件的发生也会触发特定函数被调用。

在布局编辑器中任意选中一个按钮控件，点击右键选择菜单上的View Callbacks，读者会看到一个默认的回调函数Callback和四个其他回调函数，包括CreateFcn、DeleteFcn、ButtonDownFcn和KeyPressFcn等。选择一个菜单项，MATLAB编辑/调试器会自动调到该控件对应的该回调函数上，读者可以直接在那里填写功能代码。每个回调函数都有输入参数，MATLAB对其进行了详细的说明，请读者务必认真阅读。

19.5.3 访问控件

GUI中所有控件的句柄都被放置在一个名为handles(意为所有句柄的集合)的根结构之下，读者可以通过该结构用Tag访问各个控件的句柄，例如变量列表的句柄就是handles.lst_Variables。handles会出现在任何一个回调函数之中，所以读者不用担心找不到它。

19.5.4　控件之间数据共享

控件之间传递数据也要通过定义在 handles 下的变量进行，这些变量一般在调用 OpeningFcn 的时候定义，然后它们的生存期会一直持续到程序结束。因而在各个回调函数中都可以访问它们。需要注意的是，读访问可以直接进行，但写访问必须带有强制更新的命令才能生效，否则不起作用。这类似于函数中的形参和实参的问题。比如，若要将 handles 下的变量 Time 复制到 t 中，可以输入

```
t = handles.Time;
```

但若要用 t 更新 handles 中的 Time 变量，必须输入

```
handles.Time = t;
guidata(hObject,handles);
```

其中 hObject 是这段代码所在的回调函数对应的控件。如果没有第二行语句，handles.Time 在结束这个回调函数后就会恢复成原来的值。

19.5.5　访问工作空间中的数据

访问工作空间中的数据需要用到一个特殊的函数 evalin，调用格式为

```
vars = evalin('base','who');
```

意为在名为 base 的工作空间(即默认工作空间)中执行 who 命令，并将返回的若干变量名以单元数组形式保存在 vars 中。

第六节　本例的程序和运行结果

首先定义两个子函数分别用于获取工作空间中的变量名到变量列表中，和获取变量列表中被选中的变量名，这两个函数的定义可以写在 M 文件的任意两个函数之间。

```
function RefreshVariables(handles)
vars = evalin('base','who');
set(handles.lst_Variables,'String',vars);

function var = getVariableSel(handles)
vars = get(handles.lst_Variables,'String');
idx = get(handles.lst_Variables,'Value');
var = vars{idx(1)};
```

接下来先修改 OpeningFcn 函数，在 handles 中增添若干变量用于在控件间传递数据，并且把初始值在两个编辑框中显示出来。

```
handles.Time = [];
handles.Data = [];
handles.FreqSampleNum = 100;
handles.FreqRange = [-10,10];
set(handles.ed_FreqSampleNum,'String',num2str(handles.FreqSampleNum));
set(handles.ed_FreqRange,'String',num2str(handles.FreqRange));
```

注意前边四句命令必须要写在该函数中自带的 guidata(hObject,handles);语句之前进行,否则是无效的。

然后把 RefreshVariables 添加到这里,同时也添加到 Refresh 按钮的默认回调函数之中。这样除了程序启动时,在点击这个按钮时也会更新工作空间中的变量到列表中。这时候可以运行一下程序,检验是否可以看到工作空间中的变量了。

现在修改 Set Sample Time 的默认回调函数,把当前列表中被选中的变量名赋值给 handles.Time 保存起来,添加

```
handles.Time = getVariableSel(handles);
guidata(hObject,handles);
```

同理修改 Set Sample Data 的默认回调函数以保存被选中的变量名到handles.Data 中,不再赘述。

为了让用户修改频域区间的操作生效,跳到 Frequency Range 编辑框的默认回调函数并添加

```
handles.FreqRange = str2num(get(hObject,'String'));
guidata(hObject,handles);
```

同理修改 Frequency Sample Num 的回调函数,不再赘述。

现在就可以添加实现最终功能的绘图函数了,在 Draw 按钮的回调函数中输入

```
%---Executes on button press in bt_Draw.
function bt_Draw_Callback(hObject,eventdata,handles)
%hObject handle to bt_Draw(see GCBO)
%eventdata reserved -to be defined in a future version of MATLAB
%handles structure with handles and user data(see GUIDATA)
t = evalin('base',handles.Time);
x = evalin('base',handles.Data);
plot(handles.ax_Time,t,x);
axis tight;
N = length(t);
t = reshape(t,N,1);
```

```
T = (max(t) - min(t))/(N - 1) * N;
OMGrg = str2num(get(handles.ed_FreqRange,'String'));;
OMG = OMGrg(2) - OMGrg(1);
K = str2num(get(handles.ed_FreqSampleNum,'String'));
omg = linspace(OMGrg(1),OMGrg(2) - OMG/K,K)';
FT = T/N * exp( - j * kron(omg,t.'));
X = FT * x;
plot(handles.ax_Freq,omg,abs(X));
axis tight;
```

为了提高程序的稳健性，还需要判断 handles.Time 和 handles.Data 两个变量中是否已经赋值，如果未赋值就不允许执行绘图操作，即把 Draw 按钮变灰。请读者自己阅读中国高校电工电子课程网上的程序学习，这里不再专门讲解。

现在可以自己在命令窗口中建立两个变量，例如定义

```
myt = [ - 1:0.01:1]';
myx = myt> - 0.5&myt>0.5;
```

再启动 tfviewer 程序，分别选择两个变量为抽样时间和抽样数据，按 Draw 按钮后，程序界面如图 19.6 所示。可见已经完全实现了预期的目的，为用户查看工作空间中时域信号的频谱提供了友善的 GUI。

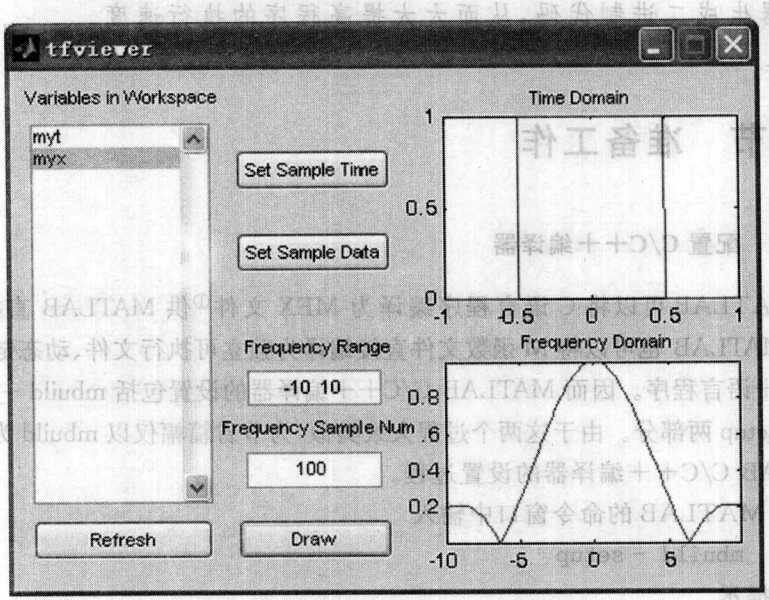


图 19.6　tfviewer 程序运行结果

第二十章　与C/C++混合编程

通过对本书的学习，读者肯定已经切身体会到MATLAB是一个功能非常强大的数学软件，从信号处理、语音处理、数值运算到电子仿真，几乎在各个工业领域，都已经得到了广泛应用，同时也取得了巨大的成功。但是，由于MATLAB是一种脚本语言，需要逐行解释执行，因而执行效率很低；另外，M文件是不能脱离MATLAB的应用程序环境而运行的，这也极大地制约了程序的可移植性和通用性。

本章将介绍MATLAB与C/C++语言混合编程的方法，这样就可以把MATLAB的强大功能融入各种应用程序中，并通过高级语言编译器生成二进制代码，从而大大提高程序的执行速度。

第一节　准备工作

20.1.1　配置C/C++编译器

MATLAB可以将C语言程序编译为MEX文件[①]供MATLAB直接调用。另外，MATLAB也可以将M函数文件直接编译为独立可执行文件、动态链接库和C/C++语言程序。因而MATLAB C/C++编译器的设置包括mbuild -setup和mex -setup两部分。由于这两个过程大致类似，为节省篇幅仅以mbuild为例介绍MATLAB C/C++编译器的设置过程。

在MATLAB的命令窗口中输入

```
mbuild -setup
```

系统将提示

① 在Windows平台下，MEX文件实质上是动态链接库。

```
Please choose your compiler for building standalone MATLAB ap-
plications:
Would you like mbuild to locate installed compilers [y]/n?
```

输入“y”并回车后MATLAB将列出当前Windows系统中安装的所有编译器并提示用户选择其中之一，系统显示

```
Select a compiler:
[1] Lcc C version 2.4.1 in D:\MATLAB71\sys\lcc
[2] Microsoft Visual C/C++ version 7.1 in C:\Program Files\
    Microsoft Visual Studio.NET 2003
[3] Microsoft Visual C/C++ version 6.0 in C:\Program Files\
    Microsoft Visual Studio

[0] None

Compiler:
```

本书将以VC/C++6.0为例，因而按“3”后回车，出现

```
Compiler:Microsoft Visual C/C++ 6.0
Location:C:\Program Files\Microsoft Visual Studio

Are these correct? ([y]/n):
```

再按“y”并回车，MATLAB将把VC/C++6.0设置为编辑器并回显过程信息。接下来在命令窗口中输入mex -setup配置MEX文件的编译器，步骤和mbuild -setup类似，不再赘述。

至此，即可调用mbuild和mex编译C/C++语言文件。以下几节详细介绍编译过程。

20.1.2　理解mxArray

MX API提供对mxArray操作的函数，用于实现mxArray和C基本数据类型之间的转换。MX API函数有一百多个，表20.1列出了其中有代表性的部分函数并解释其基本功能。请读者根据需要自行学习其他MX API函数。

表20.1　部分MEX API函数

mxAddField	为mxArray添加域
mxArrayToString	将mxArray转为字符串类型
mxCalcSingleSubscript	返回从第一个元素到特定元素的偏移地址

续表

mxCalloc	分配动态内存
mxChar	字符串类 mxArray 对应的数据类型
mxClassID	返回表明 mxArray 类的整数值
mxComplexity	判断 mxArray 是否有虚部
mxCreateDoubleMatrix	新建两维双精度浮点型 mxArray
mxDestroyArray	释放 mxCreate 类函数建立的 mxArray 的动态内存
mxDuplicateArray	复制 mxArray
mxFree	释放 mxCalloc 分配的动态内存
mxGetDimensions	返回 mxArray 维数变量的指针
mxGetField	由域名和索引得到结构中某个域的值
mxGetM	返回 mxArray 的行数
mxGetN	返回 mxArray 的列数
mxGetPr	返回指向 mxArray 的实部数据的指针
mxGetPi	返回指向 mxArray 的虚部数据的指针
mxGetString	将字符串类 mxArray 转为 C 语言的字符串
mxIsDouble	判断 mxArray 是否是双精度浮点数
mxIsEmpty	判断 mxArray 是否为空
mxSetField	由域名和索引设结构中某个域的值

第二节 从 MATLAB 中调用 C/C++程序

和 C/C++语言相比，MATLAB 的优势是功能强大，编程简单。但 C/C++语言通用性强，用其开发的程序执行效率更高，而且已经存在了大量的程序供使用，因而在某些情况下有必要在 MATLAB 中调用 C/C++语言开发的函数。在 MATLAB 中调用 C/C++等外部程序需要借助编译器将其代码编译为 MEX 文件后才能实现，其中 MEX 文件包含 MATLAB 解释器可以动态装载和执行的动态链接模块。我们称按照 MATLAB 规定的书写格式编写的 C/C++程序文件为 C MEX 文件，用 mex 命令编译该文件将生成扩展名为“.mex”的 MEX 文件。MEX 文件调用方法和一般的 M 函数文件完全相同，而且优先级比 M 文件要高，在同名的情况下，被执行的将是 MEX 文件。

本节首先介绍 C MEX 文件主要结构，然后列出全部 MEX API 以及部分 MX API 并简要说明其功能，最后举例演示 C MEX 文件的编程和编译步骤。

20.2.1　C MEX 文件结构

C MEX 文件中必须引用“mex.h”头文件。C MEX 文件的主函数不是 main 而是 mexFunction，其声明格式如下

```
#include "mex.h"
void mexFunction(int nlhs,mxArray * plhs[],int nrhs,const mx-
Array * prhs[]);
```

其中 nlhs 和 nrhs 分别是输出和输入参数的个数，plhs 和 prhs 分别是包含输出和输入参数的 mxArray。MATLAB 和 C MEX 文件交互的所有参数，无论是什么数据类型，都要以 mxArray 结构表示。可见，在已有 C 函数实现预定功能的情况下，C MEX 文件的主要任务就是完成 MATLAB 与 C 函数的数据交互问题。MATLAB 提供了一系列应用程序接口（Application Program Interface，API）函数实现 mxArray 结构和 C 语言中 int、double 等基本数据类型的转换，其中以 mx 开头的函数（称为 MX API）主要用于对 mxArray 进行操作，mex 开头的函数（称为 MEX API）提供 MATLAB 的后台操作。

20.2.2　MEX API 函数

MEX API 提供 MATLAB 环境后台操作函数，它们只能在 C MEX 文件中使用。表 20.2 列出了全部 MEX API 函数及其基本描述。请读者了解这些函数的功能范围，在需要使用时查看帮助，掌握其详细调用接口。

表 20.2　MEX API 函数

mexAtExit	MEX 文件被清除或者 MATLAB 退出时将被调用
mexCallMATLAB	调用 MATLAB 函数或用户自定义的 M 或 MEX 文件
mexErrMsgIdAndTxtIssue	显示错误信息和标示并返回 MATLAB
mexErrMsgTxt	显示错误信息并返回 MATLAB
mexEvalString	在调用者的工作空间中执行 MATLAB 命令
mexFunction	C MEX 文件的调用入口
mexFunctionName	当前 MEX 函数的函数名
mexGet	获取句柄图形的属性值
mexGetVariable	从别的工作空间复制变量
mexGetVariablePtr	从别的工作空间得到变量的只读指针
mexIsGlobal	判断某 mxArray 是否全局有效
mexIsLocked	判断某 MEX 文件是否被锁定
mexLock	锁定某 MEX 文件防止系统将其从工作空间中清除掉
mexMakeArrayPersistent	将某 mxArray 设置为永久变量，程序返回后仍存在
mexMakeMemoryPersistent	将某被分配的内存设置为永久保留，程序返回后仍存在
mexPrintf	和 ANSI C 的 printf 类似的打印输出程序

续表

mexPutVariable	将 mxArray 从 MEX 文件复制到其他工作空间
mexSet	设置句柄图形的属性值
mexSetTrapFlag	设置调用 mexCallMATLAB 出错后返回的位置
mexUnlock	允许系统从内存中清除掉 MEX 文件
mexErrMsgIdAndTxtIssue	显示警告信息和标示并返回 MATLAB
mexErrMsgTxt	显示警告信息并返回 MATLAB

20.2.3 C MEX文件实例

根据前面介绍的C MEX文件结构以及MEX API和MX API函数，举例演示C MEX文件的编写和编译方法。

例 20.1 假设有一个实现特定功能 $y=2x$ 的C函数 times2(y,x)如下

```
void times2(double y[],double x[])
{
  y[0] = 2.0 * x[0];
}
```

请编写C MEX程序将此功能模块编译为可以被MATLAB调用的MEX文件。

解 C MEX程序如下，请参考注释学习。

```
#include "mex.h"     /* 必须包含"mex.h"头文件 */

/* 假设这个C函数是待实现的功能模块，它的功能非常简单，*/
/* 对将输入的浮点数值乘以~2~后赋给输出浮点数值。 */
void times(double y[],double x[])
{
  y[0] = 2.0 * x[0];
}
/* C MEX文件的主函数 */
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,
                 const mxArray *prhs[])
{
  double *x, *y;
  int mrows,ncols;
  /* 检查输入输出参数的个数是否正确 */
  if(nrhs ! = 1)
```

```
{
  mexErrMsgTxt("One input required.");
}
else if(nlhs>1)
{
  mexErrMsgTxt("Too many output arguments");
}
/* 输入必须是一个实的标量浮点数。*/
mrows = mxGetM(prhs[0]);
ncols = mxGetN(prhs[0]);
if(! mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
    ! (mrows = = 1 && ncols = = 1))
{
  mexErrMsgTxt("Input must be a noncomplex scalar double.");
}
/* 新建保存返回值的矩阵 */
plhs[0] = mxCreateDoubleMatrix(mrows,ncols,mxREAL);
/* 取出输入和输出参数所在内存的指针 */
x = mxGetPr(prhs[0]);
y = mxGetPr(plhs[0]);
/* 用指针调用前述功能模块 */
times2(y,x);
}
```

将以上文件保存为 times2.c，然后在 MATLAB 命令窗口中输入

```
mex times2.c
dir times2.*
```

将回显

```
times2.c times2.mexw32
```

可见除了 times2.c 外，新出现了一个 times2.mexw32 文件，这就是刚编译出来的 MEX 文件，可以在命令窗口中直接调用它

```
times2(3)
ans =
     6
```

说明新建立的 MEX 文件已经将 C 程序开发的功能模型封装好，可以在 MATLAB 中直接调用。

第三节 从C/C++程序中调用MATLAB函数

用MATLAB开发数值计算和数据可视化程序方便灵活。为支持在C/C++程序中调用MATLAB函数，MATLAB把若干函数封装起来，作为引擎库供外部程序调用。这些引擎库和MATLAB一起，允许用户在外部启动MATLAB进程，创建变量和在MATLAB工作空间中进行计算。

20.3.1 引擎库API函数

表20.3列出了全部MATLAB引擎库函数及其功能说明。请读者结合帮助学习其详细使用方法。

表20.3 MEX API函数

engOpen	启动MATLAB引擎
engClose	关闭MATLAB引擎
engGetVariable	从MATLAB引擎中获取数组
engPutVariable	将数组送往MATLAB引擎
engEvalString	执行MATLAB命令
engOutputBuffer	建立一个存储MATLAB输出文本的缓冲区
engOpenSingleUse	启动一个MATLAB引擎会议做单一非共享使用
engGetVisible	检测某MATLAB引擎是否可见
engSetVisible	显示或隐藏某MATLAB引擎

20.3.2 程序实例

下面用一个详细的实例演示如何在C程序中调用MATLAB函数。本例包括两部分，第一部分把C程序中定义的双精度数组time传递到MATLAB空间并命名为变量T，然后在MATLAB空间中执行

```
D = 0.5. * ( - 9.8). * T^2;
plot(T,D);
title('Position vs. Time for a falling object');
xlabel('Time(seconds)');
ylabel('Position(meters)');
```

第二部分提示用户输入任意一句为变量X复制的MATLAB命令，C程序读到这句命令后在MATLAB空间中查询变量X的类型并打印在屏幕上。请阅读下面程序，结合注释学习各种API函数的用法。

```
#include <stdlib.h>
```

```
#include <stdio.h>
#include <string.h>
#include "engine.h"
#define BUFSIZE 256

int main( )
{
  Engine *ep;
  mxArray *T = NULL, *result = NULL;
  char buffer[BUFSIZE];
  double time[10] = {0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};
  /* 启动 MATLAB 引擎。*/
  if(!(ep = engOpen("\0")))
  {
    fprintf(stderr,"\nCan't start MATLAB engine\n");
    return EXIT_FAILURE;
  }
  /* 第一部分工作:将数据发送到 MATLAB 引擎空间,计算并绘制结果。*/
  /* 建立双精度数组 T */
  T = mxCreateDoubleMatrix(1,10,mxREAL);
  memcpy((void *)mxGetPr(T),(void *)time,sizeof(time));
  /*将变量 T 送入 MATLAB 引擎空间。*/
  engPutVariable(ep,"T",T);
  /* 距离 = (1/2)g.*t.^2,g 是重力加速度,t 是时间。*/
  engEvalString(ep,"D = .5.*(-9.8).*T.^2;");
  /* 绘制运行结果。*/
  engEvalString(ep,"plot(T,D);");
  engEvalString(ep,"title('Position vs.Time for a falling
  object');");
  engEvalString(ep,"xlabel('Time(seconds)');");
  engEvalString(ep,"ylabel('Position(meters)');");
  /* 暂停程序,用户敲任意键后再次运行。*/
  printf("Hit return to continue\n\n");
  fgetc(stdin);
  /* 第一部分工作完成,释放内存。*/
```

```
    printf("Done for Part I.\n");
    mxDestroyArray(T);
    engEvalString(ep,"close;");
    /* 第二部分工作:执行用户输入的明显。*/
    /* 申请一块缓冲区接收MATLAB命令的输出结果。*/
    engOutputBuffer(ep,buffer,BUFSIZE);
    while(result == NULL)
    {
      char str[BUFSIZE];
      /*读取用户输入的字符串。*/
      printf("Enter a MATLAB command to evaluate. This command should\n");
      printf("create a variable X. This program will then determine\
      n");
      printf("what kind of variable you created.\n");
      printf("For example:X = 1:5\n");
      printf(">>");
      fgets(str,BUFSIZE-1,stdin);
      /* 执行刚才用户输入的命令。*/
      engEvalString(ep,str);
      /* 回显MATLAB命令输出,前两个字符已被写成>>所以跳过*/
      printf("%s",buffer+2);
      /* 读取计算结果 */
      printf("\nRetrieving X...\n");
      if((result = engGetVariable(ep,"x")) == NULL)
        printf("Oops! You didn't create a variable X.\n\n");
      else{
        printf("X is class %s\t\n",mxGetClassName(result));
      }
    }
    /* 主要功能已完成,释放内存,关闭引擎。*/
    printf("Done!\n");
    mxDestroyArray(result);
    engClose(ep);
    return EXIT_SUCCESS;
}
```

20.3.3　用VC++编译程序

以VC++6.0为例介绍调用MATLAB引擎的C程序的编译方法，其他开发环境和编译器的使用方法类似，读者可自行练习。

首先启动VC++6.0集成开发环境，新建一个Win32 Console Application模板下的an empty project，然后将上一小节介绍的程序文件加入到该工程中。

然后设置链接库，在VC++6.0菜单上选择Project→Settings，弹出Project Settings对话框，选择右上方众多标签页面中的Link页面，确认Category下拉框中内容为General，对话框中部有一个Object/library modules:编辑框，其中包括了所有链接在本程序中的模块，在最后填入三个模块libeng.lib、libmx.lib和libut.lib，每个模块直接用空格分开。设置结束后对话框界面如图20.1所示。

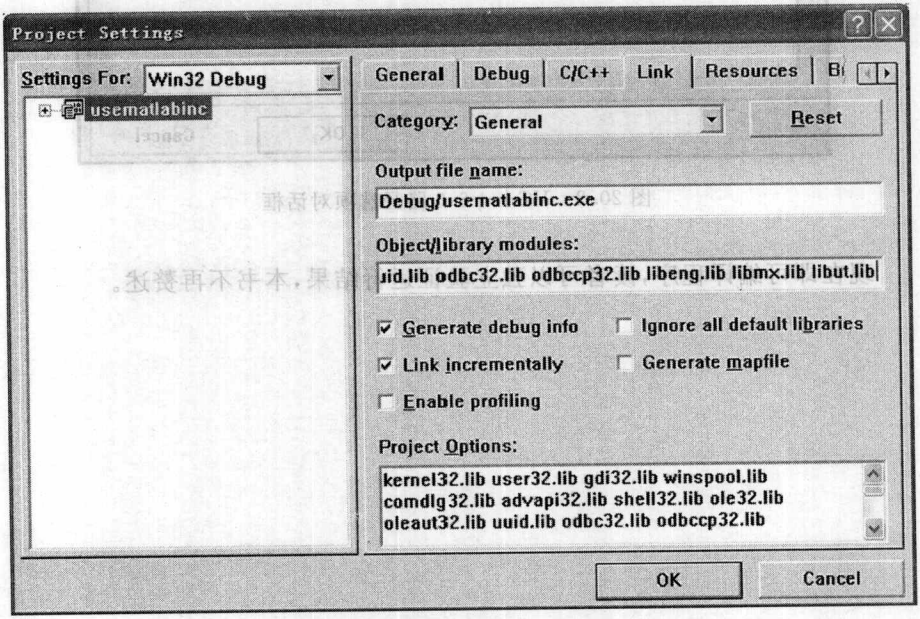


图20.1　VC++6.0程序设置对话框

最后设置include路径和library路径，在VC++6.0菜单上选择Tools→Options，弹出的Options对话框的顶部有一系列标签页面，选择Directories页面，再在右侧Show Directories for:下的下拉框中选择Include files，此时下方Directories列表中列出的即是当前编译器搜索的include路径。请将MATLABROOTPATH\extern\include路径添加该列表中，添加后的对话框界面如图

20.2所示。接下来再添加library路径，同样上述对话框，在右侧Show Directories for:下的下拉框中选择Library files，然后将MATLABROOTPATH\extern\lib\win32\microsoft路径添加到Directories列表中。

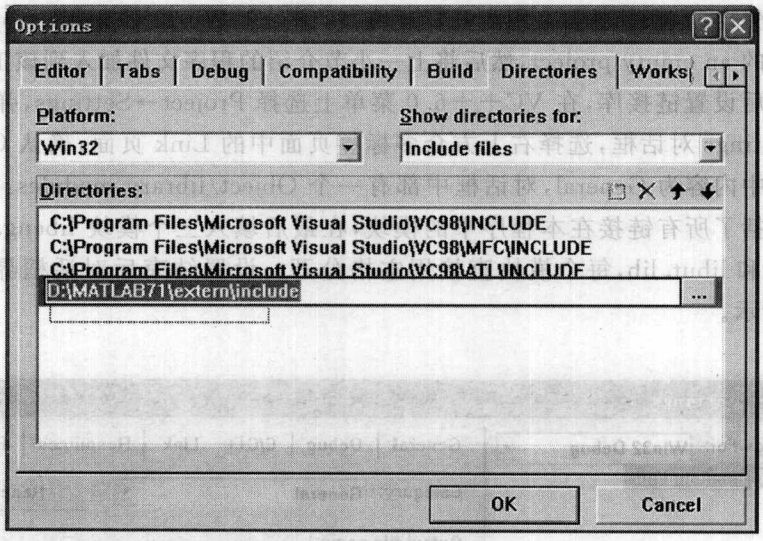


图20.2 VC++6.0路径选项对话框

现在即可编译程序，读者可以独立验证运行结果，本书不再赘述。

索　引

一、本书涉及的 MATLAB 命令、知识点及页码

addpath,202
all,8,84
area,143
auread,77
auwrite,77
avifile,77
aviread,77
axes,17

bar,143
bar3h,143
break,11
butter,151

canon,201
case,11
catch,11
cell,6,194
celldisp,194
cellplot,194
char,5
class,188
clear,10,48,64
collect,61
compass,143
compose,61
connect,189
continue,11
contour,142
contour3,143
conv,30,38,85,87
csvread,77
csvwrite,77
ctrb,203

decimate,74
deconv,30,87
diff,61
dirac,11
dlmread,77
dlmwrite,77
dsolve,61

else,11
elseif,11
end,11,124
eqtflength,94
evalin,227

ezplot,17

factor,61
fclose,126
feather,143
feedback,190
fft,97
figure,14
filter,81,84,85,165
finverse,61
fopen,126
for,11
format,167
fourier,42
frd,188
fread,126
freqs,64
freqz,101
fseek,126
fsolve,61
func2str,125
function,10,123
function_handle,6
fwrite,126

gca,18
gcbf,18
gcbo,18
gcf,18
gco,18
get,18,188
ginput,198
global,124
grpdelay,151
gtext,198
guide,222

heaviside,9,12,91
help,5
hilbert,154
hist,143
hold,14

if,11
ifft,97
ifourier,42
ilaplace,57
image,17
impulse,35
impz,84
imread,77
imwrite,77
Inf,12
input,198
int,61
interp,74
isa,6,188
isdouble,6
isfield,67
isstruct,67
iztrans,91,93

jacobian,61

keyboard,198
kron,47,165

laplace,57
latex,175
legend,15

length,9
light,17
limit,61
line,17
linsolve,61
linspace,44
load,75
logical,5
lsim,29,31
ltimodels,188
ltiprops,188

mbuild,230
mesh,143
mex,230

NaN,12
nargin,124
nargout,124
numeric,5

obsv,204
ode113,34
ode15s,34
ode23,34
ode45,34
open,11
otherwise,11

parallel,190
patch,17
path,202
pathtool,202
pause,68
persistent,124
pie3,143
plot,14
plot3,142
polar,143
pole,62
poly,30
polyder,30
polyint,30
polyval,30
polyvalm,30
psd,171
pulstran,165
pzmap,62

quiver,143

randn,171
rank,203
rectangle,17
rectpulse,165
resample,74
reshape,165
residue,58
residuez,91
return,198
rlocfind,194
rlocus,194
rmpath,202
roots,28,30,62
rose,143

sawtooth, 52
series,190
set,18,188
setstr,205

simple,61
simplify,61
simulink,127
solve,61
sound,74
spectrogram,221
sprintf,205
square,52,165
ss,27,188,207
ss2ss,200
ss2tf,188
ss2zp,188
ssdata,188
stairs,143
stem,17
stem3,143
step,35
stepz,84
str2func,125
str2num,205
strcat,205
struct,6,66
subplot,15
subs,9,61
surf,142
surface,17
switch,11
sym,6,9
syms,9

text,17
textscan,77
tf,26,61,188
tf2ss,188
tf2zp,62,94,188
tf2zpk,188
tfdata,189
title,15
tripuls,163
try,11

waterfall,142
wavread,77
wavwrite,77
while,11
who,75,166

xcorr,171
xlabel,15
xlsread,77
xlswrite,77

ylabel,15

zero,62
zeros,9
zp2tf,62,94,188
zpk,188
zplane,94,101
ztrans,90

二、插入各章节的MATLAB知识点及页码

多项式,30
解微分方程(组),33

是否需要返回值,37
连续时间信号的数值计算,40
程序优化技巧,47
生成周期性连续信号,51
提高函数的稳健性,54
符号运算函数,61
虚数单位 i,j 及常量恢复,63
结构,66
访问标准格式文件,76
处理列矢量,84
多项式相乘和卷积的关系,87
命令和行的关系,96
交互式信号处理工具,105
低级文件访问,119
生成常用信号,165
数字显示格式,167
图形窗口显示数学符号,175
函数和运算符重载,190
单元数组,194
用户交互操作,198
搜索顺序和搜索路径,202
字符串,204

参考文献

[1] 郑君里，应启珩，杨为理. 信号与系统(上册)[M]. 2 版. 北京：高等教育出版社，2000.

[2] 郑君里，应启珩，杨为理. 信号与系统(下册)[M]. 2 版. 北京：高等教育出版社，2000.

[3] 郭文彬，桑林. 通信原理——基于 MATLAB 的计算机仿真[M]. 北京：北京邮电大学出版社，2006.

[4] 杨行峻，迟惠生，等. 语音信号数字处理[M]. 北京：电子工业出版社，1995.

[5] 陈怀琛，吴大正，高西全. MATLAB 及在电子信息课程中的应用[M]. 3 版. 北京：电子工业出版社，2006.

[6] Kirk Donald E，Strum Robert Denney. 现代线性系统——使用 MATLAB [M]. 刘树棠译. 西安：西安交通大学出版社，2002.

[7] 郑君里. 教与写的记忆——信号与系统评注[M]. 北京：高等教育出版社，2005.

[8] 孙祥，徐流美，吴清. MATLAB 7.0 基础教程[M]. 北京：清华大学出版社，2005.

[9] 梁虹，普园媛，梁洁. 信号与线性系统分析——基于 MATLAB 的方法与实现[M]. 北京：高等教育出版社，2006.

[10] Bishop Robert H. Modern Control Systems Analysis and Design Using MATLAB and Simulink(现代控制系统分析与设计——应用 MATLAB 和 Simulink)[M]. 影印版. 北京：清华大学出版社，2003.

[11] Proakis John G，Ingle Vinay K. 数字信号处理——使用 MATLAB[M]. 刘树棠译. 西安：西安交通大学出版社，2002.

[12] Salehi Masoud，Proakis John G. 现代通信系统——使用 MATLAB[M]. 刘树棠译. 西安：西安交通大学出版社，2001.

[13] Singer Andrew C，Buck John R，Daniel Michael M. 信号与系统计算机练习——利用 MATLAB[M]. 刘树棠译. 西安：西安交通大学出版社，2002.

[14] Bradley Kevin，Stonick Virginia. Labs for Signals and Systems Using MATLAB[M]. Boston：PWS Publishing Company，1996.