MATLAB 仿真与应用系列丛书

详解 MATLAB 数字信号处理

张德丰　编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书首先介绍 MATLAB 的发展史及影响、MATLAB 的基本运算等，使读者对 MATLAB 有一个概略的了解。然后系统论述数字信号处理的基本概念、工作原理及在工程中的算法。精选科学和工程计算中常用的多个算法，全部采用了 MATLAB 语言编程实现，并结合实例对算法程序进行验证和分析。其中详细讲解了信号的分析基础、系统模型及数据采集分析、信号的变换、模拟滤波器、IIR 滤波器设计、FIR 滤波器、MATLAB 的其他滤波器、随机信号及参数建模、信号在小波分析与处理中的应用等，最后举例介绍 MATLAB 在数字信号中的具体应用。

本书可作为通信、电子等相关专业高年级本科生和研究生的学习用书，也可供从事数字信号处理的工程设计人员参考。

图书在版编目（CIP）数据

详解 MATLAB 数字信号处理 / 张德丰编著.—北京：电子工业出版社，2010.6
（MATLAB 仿真与应用系列丛书）

ISBN 978-7-121-10994-2

Ⅰ. ①详… Ⅱ. ①张… Ⅲ. ①数字信号－信号处理－计算机辅助计算－软件包，MATLAB Ⅳ. ①TN911.72

中国版本图书馆 CIP 数据核字（2010）第 100235 号

责任编辑：陈韦凯
特约编辑：李玉昌
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：23　字数：588 千字
印　　次：2010 年 6 月第 1 次印刷
印　　数：4000 册　　定价：45.00 元

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

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

服务热线：（010）88258888。

前　言

MATLAB 最初主要用于矩阵数值的计算，随着它的版本的不断升级，其功能越来越强大，应用范围也越来越广阔。如今，MATLAB 已经发展成为国际上非常流行的科学与工程计算语言之一，它使用方便、输入简捷、运算高效、内容丰富，是高等院校理工科教学和科研中常用且必不可少的工具之一，掌握 MATLAB 已经成为相关专业本科生、研究生和教师的必备技能。

MATLAB 是一种工程计算的高级语言。美国的 MathWorks 公司自 1984 年推出它的 DOS 版本后，又推出了它的 Windows 版本，并且不断推出更新的版本，使得 MATLAB 的涵盖领域越来越广，到目前为止，已经有仿真工具 Simulink 及其他如自动控制、信号处理、图像处理、神经网络、模式识别、小波分析、数理统计、生物信息等 30 多个工具箱。由于其灵活的编程方法和极高的编程效率，加上其在用户界面和功能上的不断扩展，自推出以来，日益受到广大高校师生和科研人员的青睐。

本书是作者结合数字信号处理理论和 MATLAB 操作技术，提供给读者的一本实践性很强的工具书。本书介绍数字信号处理基本原理的同时，非常重视信号处理的实现问题，对所有例子都给出了具体实现的 MATLAB 程序。把理论与仿真实验结合在一起，既突出了理论的物理概念，又使读者能在实践中掌握数字信号处理的基本概念、基本方法和基本应用，达到学以致用的目的，起到事半功倍的效果。

本书共分为 11 章。第 1 章是 MATLAB 概述，介绍 MATLAB 的发展史及影响，MATLAB 的基本运算等，使读者对 MATLAB 有一个概略的了解；第 2 章介绍信号的分析基础，如时间信号及采样定理、连续时间信号在 MATLAB 中的运算、离散时间信号在 MATLAB 中的运算等；第 3 章介绍系统模型及数据采集分析，包括系统数学模型、数据采集过程等内容；第 4 章介绍信号的变换，涉及 Z 变换、离散傅里叶变换、快速傅里叶变换等内容；第 5 章介绍模拟滤波器，包括模拟滤波器的基本概念、模拟滤波器的原型设计等内容；第 6 章介绍 IIR 滤波器设计，包括 IIR 滤波结构、常用模拟低通滤波器特性等；第 7 章介绍 FIR 滤波器，包括 FIR 滤波器的结构、线性相位 FIR 数字滤波器的特性等内容；第 8 章介绍 MATLAB 的其他滤波器，如自适应滤波器、格型滤波器、线性预测滤波器；第 9 章介绍随机信号及参数建模，包括随机信号基本处理、随机信号的相关函数和协方差等内容；第 10 章是小波分析在信号处理中的应用，包括信号的小波变换、信号重构、信号分析等内容；第 11 章总结介绍 MATLAB 在数字信号中的应用。

本书结构紧凑，仿真示例丰富，同时力求图文并茂，文字流畅，使之成为学习和使用 MATLAB 进行数字信号处理仿真研究的有价值的参考书之一。当然，在编写的过程中，错误或疏漏之处在所难免，敬请各位读者批评指正。

本书可作为高等工科院校通信、电子信息、计算机、信息工程、自动控制等相关专业的本科及研究生教材，也可供从事信号处理相关工作的科技工作者参考。

本书主要由张德丰负责编写。参与图书编写及源程序校对、调试等工作的还有雷小平、周燕、周灵、崔如春、李娅、栾颖、刘志为和周品等。

为便于读者学习，本书免费提供程序的源代码，读者可通过登录华信教育资源网（www.hxedu.com.cn）查找本书下载。

编著者

2010 年 3 月

目　　录

第 1 章　MATLAB 概述

1.1　MATLAB 简介

1.1.1　MATLAB 的发展史及影响

MATLAB 名字由 MATrix 和 LABoratory 两词的前三个字母组合而成。那是 20 世纪 70 年代后期的事：时任美国新墨西哥大学计算机科学系主任的 Cleve Moler 教授出于减轻学生编程负担的动机，为学生设计了一组调用 LINPACK 和 EISPACK 库程序的“通俗易用”的接口，此即用 FORTRAN 编写的萌芽状态的 MATLAB。经几年的校际流传，在 Little 的推动下，由 Little、Moler、Steve Bangert 合作，于 1984 年成立了 MathWorks 公司，并把 MATLAB 正式推向市场。从这时起，MATLAB 的内核采用 C 语言编写，而且除原有的数值计算能力外，还新增了数据图视功能。

MATLAB 以商品形式出现后，仅短短几年，就以其良好的开放性和运行的可靠性，使原先控制领域里的封闭式软件包（如英国的 UMIST、瑞典的 LUND 和 SIMNON、德国的 KEDDC）纷纷淘汰，而改以 MATLAB 为平台加以重建。在进入 20 世纪 90 年代时，MATLAB 已经成为国际控制界公认的标准计算软件。到 90 年代初期，在国际上三十几个数学类科技应用软件中，MATLAB 在数值计算方面独占鳌头，而 Mathematica 和 Maple 则分居符号计算软件的前两名。Mathcad 因其提供计算、图形、文字处理的统一环境而深受中学生欢迎。

MathWorks 公司于 1993 年推出 MATLAB 4.0 版本，从此告别 DOS 版。4.x 版在继承和发展其原有的数值计算和图形可视能力的同时，出现了以下几个重要变化：

（1）推出了 SIMULINK。这是一个交互式操作的动态系统建模、仿真、分析集成环境。它的出现使人们有可能考虑许多以前不得不做简化假设的非线性因素、随机因素，从而大大提高了人们对非线性、随机动态系统的认知能力。

（2）开发了与外部进行直接数据交换的组件，打通了 MATLAB 进行实时数据分析、处理和硬件开发的道路。

（3）推出了符号计算工具包。1993 年 MathWorks 公司从加拿大滑铁卢大学购得 Maple 的使用权，以 Maple 为“引擎”开发了 Symbolic Math Toolbox 1.0。MathWorks 公司此举加快结束了国际上数值计算、符号计算孰优孰劣的长期争论，促成了两种计算的互补发展新时代。

（4）推出了 Notebook。MathWorks 公司瞄准应用范围最广的 Word，运用 DDE 和 OLE，实现了 MATLAB 与 Word 的无缝连接，从而为专业科技工作者创造了融科学计算、图形可视、文字处理于一体的高水准环境。

1997 年仲春，MATLAB 5.0 版问世，紧接着是 5.1 版、5.2 版，以及 1999 年春的 5.3 版。与 4.x 版相比，现今的 MATLAB 拥有更丰富的数据类型和结构、更友善的面向对象、更加快速精良的图形可视、更广博的数学和数据分析资源、更多的应用开发工具。诚然，到 1999 年年底，Mathematica 也已经升级到 4.0 版，它特别加强了以前欠缺的大规模数据处理能力。Mathcad

也赶在 2000 年到来之前推出了 Mathcad 2000，它购买了 Maple 内核和库的部分使用权，打通了与 MATLAB 的接口，从而把其数学计算能力提高到专业层次。但是，就影响而言，至今仍然没有一个别的计算软件可与 MATLAB 匹敌。在欧、美大学里，诸如应用代数、数理统计、自动控制、数字信号处理、模拟与数字通信、时间序列分析、动态系统仿真等课程的教科书都把 MATLAB 作为内容。这几乎成了 20 世纪 90 年代教科书与旧版书籍的区别性标志。在那里，MATLAB 是攻读学位的大学生、硕士生、博士生必须掌握的基本工具。在国际学术界，MATLAB 已经被确认为准确、可靠的科学计算标准软件。在许多国际一流学术刊物上（尤其是信息科学刊物），都可以看到 MATLAB 的应用。在设计研究单位和工业部门，MATLAB 被认作进行高效研究、开发的首选软件工具。如美国 National Instruments 公司信号测量、分析软件 LabVIEW，Cadence 公司信号和通信分析设计软件 SPW 等，或者直接建筑在 MATLAB 之上，或者以 MATLAB 为主要支撑。又如 HP 公司的 VXI 硬件，TM 公司的 DSP，Gage 公司的各种硬卡、仪器等都接受 MATLAB 的支持。之后陆续推出了几个改进和提高的版本，2004 年 9 月，正式推出 MATLAB Release14，即 MATLAB 7.0，其功能在原有的基础上又有了进一步的改进；2009 年 3 月，推出 R2009a，本书即基于此版本，对其他版本软件亦适用。

1.1.2 MATLAB 的功能特点

MATLAB 的应用范围非常广，包括信号和图像处理、通信、控制系统设计、测试和测量、财务建模和分析及计算生物学等众多应用领域。附加的工具箱（单独提供的专用 MATLAB 函数集）扩展了 MATLAB 环境，以解决这些应用领域内特定类型的问题。MATLAB 的功能特点主要有如下 6 点。

1．大量引入图形用户界面

MATLAB 改变了过去单调依靠“在指令窗通过纯文本形指令进行各种操作”面貌，引入了许多让使用者一目了然的图形界面，如在线帮助的交互型界面 helpwin、管理工作内存的 workspace、交互式的路径管理界面 pathtool、指令窗显示风格设置界面等。它们的开启方式有工具条图标开启、选择菜单项开启、直接“文本式”指令开启。

2．引入了全方位帮助系统

“临场”在线帮助，这些帮助内容，大多嵌附在 M 文件中，即时性强，反应速度快。它对求助内容的回答最及时准确。MATLAB 旧版就一直采用这种帮助系统，并深受用户欢迎。新版保留原功能的同时，还新增一个内容与之完全对应的图形界面 helpwin，加强了对用户的向导。综合型在线帮助文库 helpdesk：该文库以 HTML 超文本形式独立存在。整个文库按 MATLAB 的功能和核心内容编排，系统性强，且可以借助“超链接”方便地进行交叉查阅。但是，这部分内容偶尔发生与真实 M 文件脱节的现象。完整易读的 PDF 文档：这部分内容与 HTML 帮助文库完全对应。PDF 文档不能直接从指令窗中开启，而必须借助 Adobe Acrobat Reader 软件阅读。这种文件的版面清楚、规范，适宜有选择地系统阅读，也适宜于制作硬拷贝。演示软件 demo：这是一个内容广泛的演示程序。MATLAB 一向重视演示软件的设计，因此，无论 MATLAB 旧版还是新版，都随带各自的演示程序，只是，新版内容更丰富了。

3．M 文件编辑、调试的集成环境

新的编辑器有十分良好的文字编辑功能。它可采用色彩和制表位醒目地区分标识程序中不同功能的文字，如运算指令、控制流指令、注释等。通过编辑器的菜单选项可以对编辑器的文字、段落等风格进行类似 Word 那样的设置。从 5.2 版起，还新增了“变量现场显示”功能，

只要把鼠标放在变量名上（Mouse over），就能在现场显示该变量的内容。

在 5.x 版以后的版本中，调试器已经被图形化，它与编辑器集成为一体。只需点动交互窗上的调试图标就可完成对程序的调试。

4．M 文件的性能剖析

调试器只负责 M 文件中语法错误和运行错误的定位，而性能剖析指令 profile 将给出程序各环节的耗时分析报告。5.3 版以后的版本中，剖析指令的分析报告特别详细，它将帮助用户寻找影响程序运行速度的“瓶颈”所在，以便改进。

5．Notebook 新的安装方式

从 4.2c 版引入 Notebook 以来，这种集文字、计算、图形于一体的“活”环境就深受用户赞赏。但直到 5.2 版，Notebook 的安装都是与 MATLAB 的安装同步进行的。这种安装方式的不便之处是：一旦 Word 发生变动，就必须把 MATLAB 全盘重装。5.3 版以后的版本都改变了这种局面，它可以在 MATLAB 指令窗中“随时”进行安装 Notebook，省时灵活。

6．MATLAB 环境可运行文件的多样化

旧版中，用户可编制和运行的程序文件只有 M 脚本文件和 M 函数文件。5.x 版以后的版本都新增了产生伪代码 P 文件的 pcode 指令和产生二进制 MEX 文件的 mex 指令。较之 M 文件，这两种文件的运行速度要快得多，保密性也好。

1.1.3　MATLAB R2009a 的新特点

Mathworks 公司于 2009 年 3 月发布了 MATLAB R2009a。相比以前版本而言，MATLAB R2009a 不仅包括 MATLAB 和 Simulink 的新特性，同时还包含 81 个其他产品模块的升级和 bug 修正。

从 MATLAB R2009a 开始，MATLAB 和 Simulink 产品家族软件在安装后需要激活才能使用。MATLAB R2009a 将引入 License Center——在线 License 管理的工具。

MATLAB R2009a 新版本中，产品模块进行一些调整，MATLAB Builder for COM 的功能集成到 MATLAB Builder for .net 中去，Finacial Time Series Toolbox 的功能集成到 Financial Toolbox 中。下面对 MATLAB 的新特点作简单介绍。

（1）MATLAB 产品新特性：

① MATLAB 中采用先进的面向对象编程，包括对类和对象、继承、方法、属性、事件和包的完全支持。

② Optimization Toolbox 中针对大量数据优化问题对内部点求解器和并行计算提供支持。

③ Financial Toolbox 均方差投资优化的线性互补程序。

④ Parallel Computing Toolbox 对 PBS Pro 和 TORQUE 规划的支持。

⑤ Statistics Toolbox 中交叉确认、特性选择、半随机数和并行最客服乘特性。

（2）Simulink 产品新特性：

① Simulink 中重新设计的多平台库浏览器。

② Real-Time Workshop Embedded Coder 中生成对 AUTOSAR 兼容代码。

③ Embedded MATLAB 中 M-Lint 代码分析仪和 Simulink Design Verifier 对 Embedded。MATLAB 语言子集函数生成代码进行检查。

④ Simulink Verification and Validation 提供对安全关键系统 IEC 61508 设计规则检查。

⑤ Simulink Fixed Point 提供对浮点模型的自动定点转换的指导意见。

⑥ Communication Blockset 针对调制、解调、编码和解码函数的定点支持。

⑦ Embedded IDE Link MU 作为新产品将 Simulink 模型生成代码并应用到 Green Hills MULTI 开发环境中。

⑧ 从 MATLAB R2008a 开始将不再支持 PowerPC 处理器上运行 Macintosh OS X 操作系统，也不支持 Microsoft Windows 2000 操作系统。此外，在 MATLAB R2008a 中 15 个产品模块被重新命名。

1.2 MATLAB 的用户界面

在默认情况下，图 1-1 为启动 MATLAB 时的默认工作界面。工作界面中包含几个非常重要的工作窗口，如命令窗口、工作内存窗口、命令历史窗口、当前目录窗口和图形窗口等。下面将对几个常用窗口的功能及使用进行介绍。

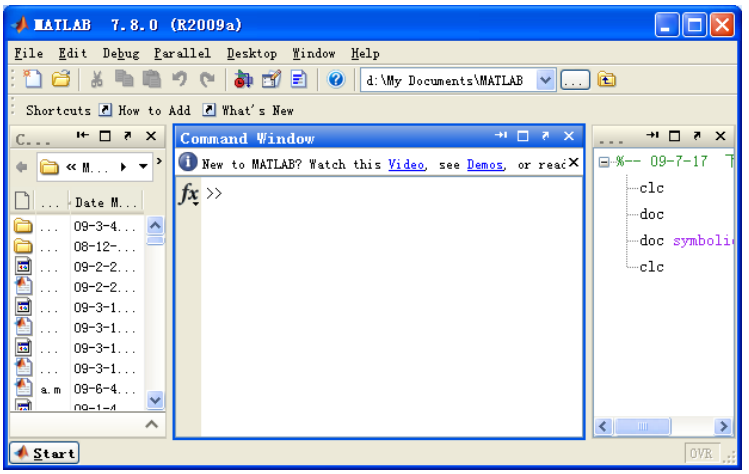


图 1-1 MATLAB 工作界面

1.2.1 MATLAB 命令窗口

在默认设置下，“Command Window（命令窗口）”自动显示于 MATLAB 界面右侧，如果用户只想调出“Command Window”，也可以单击【Desktop】菜单下【Desktop Layout】命令项中的【Command Window Only】选项。

命令窗口是和 MATLAB 编译器连接的主要窗口。“fx >>”为运算提示符，是 MATLAB R2009a 特有的提示符。表示 MATLAB 处于准备状态。MATLAB 具有良好的交互性，当在提示符后输入一段正确的运算式时，只需要按“Enter”键，命令窗口就会直接显示运算结果。

【例 1-1】矩阵 $\boldsymbol{I}=\begin{bmatrix}1 & 2 & 4\\ 3 & 4 & 5\\ 7 & 8 & 9\end{bmatrix}$ 的输入。

矩阵输入后，显示的单独命令窗口效果如图 1-2 所示。

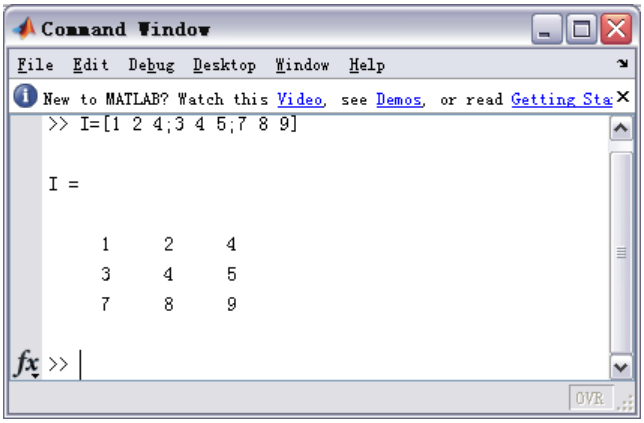


图 1-2　命令窗口显示效果

1.2.2　MATLAB 命令历史窗口

在默认设置下，“Command History（命令历史）”窗口自动显示于 MATLAB 界面左下侧，用户也可以通过单击【Desktop】菜单栏下的【Command History】命令项，调出或隐藏该窗口。

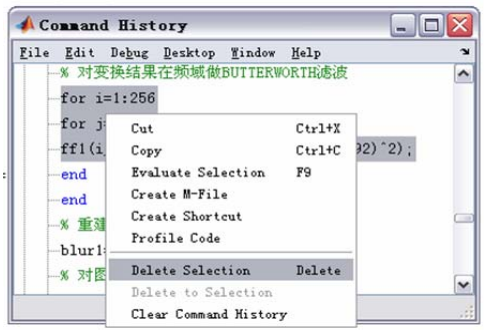


图 1-3　命令历史窗口

命令历史窗口显示用户在命令窗口中所输入的每条命令的历史记录，并标明使用时间，这样可以方便用户查询。如果用户要再次执行某条已经执行过的命令，只需在命令历史窗口中双击该命令即可；如果用户需要从命令历史窗口中删除一条或多条命令，只需选中这些命令，右击，在弹出的快捷菜单中单击【Delete Selection】命令（删除选择）即可，其窗口形式如图 1-3 所示。

1.2.3　MATLAB 工作内存浏览器窗口

在默认设置下，“Workspace（工作内存浏览器）”窗口自动显示于 MATLAB 界面左上侧，用户也可以单击【Desktop】菜单项下的【Workspace】命令，调出或隐藏该窗口。

工作内存浏览器是 MATLAB 的重要组成部分，例如，表达式 x=20 产生了一个名为 x 的变量，而且这个变量被赋予 20 的值，这个值就被存储在计算机的内存中。工作内存浏览器就是用来显示当前计算机内存中的 MATLAB 变量的名称、数学结构、该变量的字节数及其类型，如图 1-4 所示。

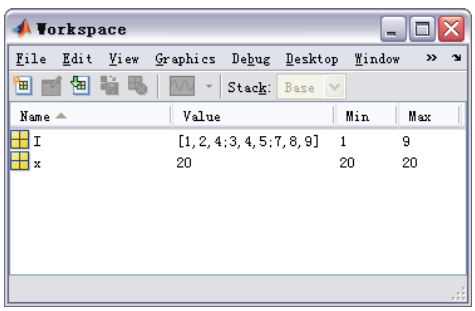


图 1-4　工作内存浏览器

MATLAB 提供了一些管理和查看内存变量的命令，通过这些命令可以方便地完成保存、加载、删除及查看当前工作空间中的变量等操作。

1．save 命令

MATLAB 中保存变量的基本命令是 save，使用该命令将当前工作空间中的变量以二进制的形式存储到后缀名为 MAT 的数据文件中，调用格式有如下 3 种：

save：将工空间中的所有变量数据都存放到名为 MATLAB.mat 的二进制 MAT 数据文件中。

save FileName：将工作空间的所有变量都保存到文件名为 FileName 的二进制 MAT 数据文件中。

save FileName 变量 1 变量2…参数：以参数指定的保存方式将工作空间中的指定变量 1 和变量 2 等保存到文件名为 FileName 的 MAT 数据文件中。参数有“-ASCII”、“-APPEND”等方式，参数可以省略。例如，指定参数为“-ASCII”则将变量 1 和变量 2 以 ASCII 文件的形式保存到名为 FileName 的 MAT 数据文件中。

2．load 命令

读取存储在文件中的变量可以使用 load 命令，该命令的使用方法与 save 类似，调用格式如下：

load：将 MATLAB.mat 数据文件中的变量加载到当前工作空间。

load FileName 变量 1 变量 2…：将指定文件中的指定变量 1、变量 2 等加载到当前工作空间，变量可以省略，省略时则加载指定文件中的所有变量。

3．查看变量的命令

除了 save 命令和 load 命令，MATLAB 还提供了如下一些常用的查看变量的命令：

① who：查看 MATLAB 内存变量变量名。

② whos：查看 MATLAB 内存变量名、大小、类型和字节数。

③ clear：删除工作空间中的变量。

MATLAB 提供判断变量是否存在于工作空间的函数，调用格式如下：

i=exist('X')：查询工作空间中是否存在变量 X，通过返回值的不同可以得到不同的信息。返回值的具体含义见表 1-1。

表 1-1 返回值的具体含义

返 回 值	含 义
i=0	表示不存在以下变量和文件
i=1	表示存在一个变量名为“X”的变量
i=2	表示存在一个名为“X.m”的文件
i=3	表示存在一个名为“X.mex”的文件
i=4	表示存在一个名为“X.md”的文件
i=5	表示存在一个名为“X”的内部函数

1.2.4 MATLAB 的当前目录窗口

当前目录浏览器窗口（Current Directory Browser）在默认情况下位于工作界面的左上方。单独显示的情况如图 1-5 所示。

1．当前目录的设置

如果是通过单击桌面上的 MATLAB 图标启动，则启动后的默认当前目录是“MATLAB/work”；如果 MATLAB 是由单击“MATLAB/bin/win32”目录下的“MATLAB.exe”启动，则默认当前目录是“MATLAB/bin/win32”。

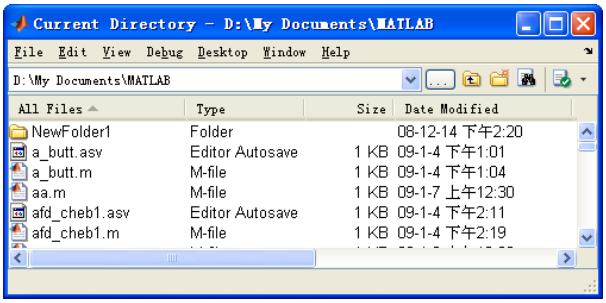


图 1-5　独立的当前目录浏览器

用户可以通过两种方法更改当前目录：一种是在当前目录设置区进行设置；另一种通过命令设置。在图 1-5 中或 MATLAB 界面工具栏的右边都有当前目录设置区，在“设置栏”中直接填写待设置的目录名即可。MATLAB 提供了修改当前目录的命令 cd，其调用格式如下：

```
cd          %显示当前目录
cd 目录     %指定当前目录
cd ...      %指定上一级目录为当前目录
```

2．M 或 MAT 文件描述区

MATLAB R2009a 默认情况下是不显示 M 或 MAT 文件描述区的，如图 1-5 所示。如果要显示 M 或 MAT 文件描述区，单击【File】菜单下的【preferences】菜单项，在弹出的“preferences”对话框中单击左侧的【Current Directory】选项，在右边【Brower Display Options】中选择【Show M-file Comments and MAT-file Comments】复选框，然后单击【OK】按钮。

3．MATLAB 搜索路径的扩展和修改

1）何时需要修改搜索路径

假如用户有多个目录需要修改，同时与 MATLAB 交换信息，或经常需要与 MATLAB 交换信息，就应该把这些目录放置在 MATLAB 的搜索路径中，使得这些目录上的文件或数据能被调用。又假如其中某个目录需要用来存放运行过程中产生的文件和数据，还应该把这个目录设置为当前目录。

2）利用“Set Path（设置路径）”对话框修改搜索路径

采用以下任何一种方法都可以调出“Set Path”对话框，如图 1-6 所示。

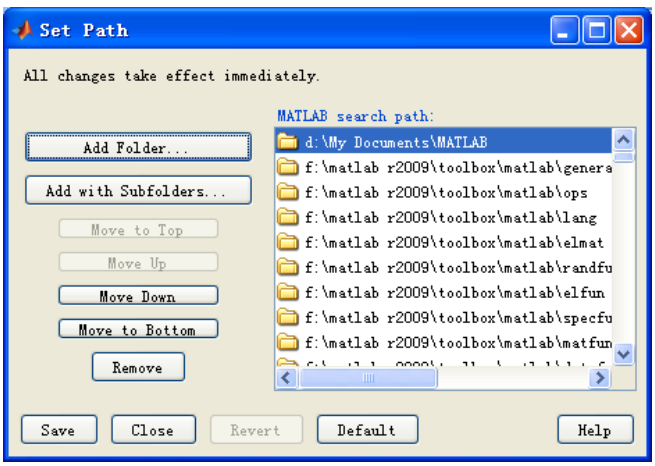


图 1-6　“Set Path”对话框

（1）在命令窗口中，运行 pathtool 命令。

（2）在 MATLAB 主窗口、命令窗口等的菜单中，单击【File】菜单下的【Set Path】选项。

注意

该对话框设置搜索路径有两种修改状态：

① 当前修改有效：假如在路径设置过程中，仅使用了该对话框的左侧按钮。

② 永久修改有效：假如在设置后，单击对话框下方的 Save 按钮。永久性修改是指所进行的修改不会因 MATLAB 的关闭而消失。

3）利用命令 path 设置路径

利用 path 命令设置路径的方法对任何版本的 MATLAB 都适用。假设待纳入搜索路径的目录为 d:\ MATLAB_file，那么以下任何一条命令均能实现：

```
>> path(path,'d:\ MATLAB_file')     %把 d:\ MATLAB_file 设置在搜索路径的尾端
>> path('d:\ MATLAB_file',path)     %把 d:\ MATLAB_file 设置在搜索路径的首端
```

用 path 命令扩展的搜索路径仅在当前 MATLAB 环境下有效。也就是说，若用户退出当前 MATLAB 后，再重新启动 MATLAB，那么在前一环境下用 path 所定义的扩展搜索路径无效。

1.2.5 MATLAB 的 M 编辑窗口

MATLAB 提供了一个内置的具有编辑和调试功能的程序编辑器。有 3 种方式可以进入程序编辑器：

（1）单击【File】菜单下的【New】选项或【Open】选项。

（2）单击工具栏中的【New】按钮或【Open】按钮。

（3）在命令编辑区中输入 eidt 命令。

MATLAB 的程序编辑器如图 1-7 所示。

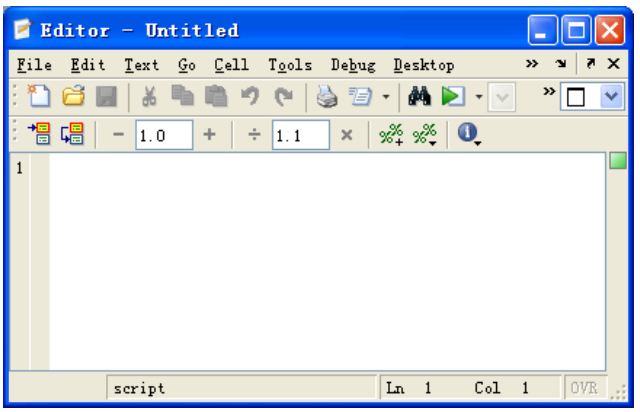


图 1-7 MATLAB 的程序编辑器

下面简单介绍 Editor 和脚本编写。

1. Editor M 文件编辑器简介

对于比较简单的问题和一次性问题，通过在命令窗口中直接输入一组命令去求解，也许是比较简捷的。但当要解决的问题所需的命令较多和所用命令结构较复杂时，或当一组命令通过改变少量参数就可以被反复使用去解决不同问题时，直接在命令窗口中输入命令的方法变显得

烦琐和笨拙。M 脚本文件就是用来解决这个问题的。

默认情况下，M 文件编辑器（Editor）不随 MATLAB 的启动而开启，只有编写 M 文件时才启动。M 编辑器不仅可以编辑 M 文件，而且可以对 M 文件进行交互式调试；不仅可以处理有.m 扩展名的文件，而且还可以阅读和编辑其他 ASCII 码文件。

2．M 脚本文件编写初步

M 脚本文件是指该文件中的命令形式和前后位置。与解决同一个问题时在命令窗口中输入的那组命令没有任何区别；MATLAB 在运行这个脚本时，只是简单地从文件中读取一条命令，送到 MATLAB 中去执行；与在命令窗口中直接运行命令一样，脚本文件运行产生的变量都是驻留在 MATLAB 基本空间中的，文件扩展名为.m。

编写 M 脚本文件，并运行此脚本文件。

例如，试绘制对应于正弦波信号的单位标准差对称误差值长条图。

其实现的 MATLAB 程序代码如下：

```
>> x=linspace(0,2*pi);
>> y=sin(x);
>> e=std(y)*ones(size(x));        %单位标准差
>> errorbar(x,y,e,'d');
>> set(gcf,'MenuBar','none','Position',[400,350,350,250]);
```

选中程序单击右键，在弹出的菜单中选择【Evaluate Selection】选项，即可在图形窗口中看到如图 1-8 所示的曲线。

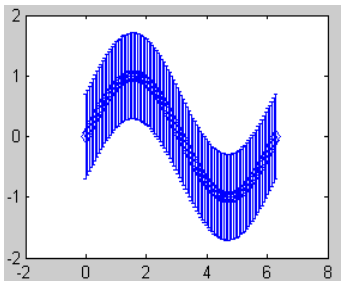


图 1-8　绘制函数曲线图像示例

1.3　变量及赋值

1.3.1　标识符与数据格式

标识符是标志变量名、常量名、函数名和文件名的字符串的总称。在 MATLAB 中，变量和常量的标识符最长允许 32 个字符。字符包括全部的英文字母（大小写共 52 个）、阿拉伯数字和下划线等符号，标识符中第一个字符必须是英文字母。

在其他计算机语言中，通常设有多种数据格式，如字符型（8 位）、整数型（16 位）等，可节省内存和提高速度，但增加了编程的复杂性。MATLAB 省去了多种数据格式，内部只有一种数据格式，那就是双精度格式，对应于 64 位二进制数据，这对绝大多数工程计算是足够的。MATLAB 可简化编程，但在运算速度和内存消耗方面付出了代价。

1.3.2　矩阵及其元素的赋值

赋值就是把数赋给代表常量或变量的标识符。赋值语句的一般形式如下：

变量=表达式（或数）

在 MATLAB 中，变量都代表矩阵，其阶数为 $n \times m$，即该矩阵共有 n 行 m 列。列向量可被当作只有一列的矩阵（$n \times 1$）；行向量（或一维数组）可被当作只有一个行的矩阵（$1 \times m$）；标量（或常量）应看作 1×1 阶的矩阵。

1．赋值要求

在输入矩阵时，应遵循以下规则：

① 整个矩阵的值应放在方括号中。

② 同一行中各元素之间以逗号“,”或空格分开。

③ 不同行的元素以分号“;”隔开。

例如，在 MATLAB 的命令窗口中输入：

```
>> A=[1,4,7,2,5,8]    %可当作一个行向量(或一维数组)
```

回车后，输出如下：

```
A =
     1     4     7     2     5     8
```

因此，变量 A 是 1×6 阶矩阵。

又如输入语句：

```
>> B=[1 4 7;2 5 8]   %注意分号“;”的功能与逗号“,”功能区别
```

输出如下：

```
B =
     1     4     7
     2     5     8
```

即变量 B 是 2×3 阶矩阵。

再如利用表达式赋值：

```
>> x=[-2.5*3,(1+2+4)/5,sqrt(3)]
```

输出如下：

```
x =
   -7.5000    1.4000    1.7321
```

如果不希望显示处理结果，可以在语句结尾加上分号“;”，这在编写 M 文件时非常有用。例如，对常量 c 赋值：

```
>> c=6;    %注意分号";"在不同位置的功能区别
```

按回车键后，将不显示结果，但已完成对变量 c 的赋值。这时若在命令窗口中输入：

```
>> c
```

输出如下：

```
c =     6
```

2．变量的元素的标注

在 MATLAB 中，变量的元素（矩阵元）用圆括号“()”中的数字（也称为下标）来标明，一维矩阵（也称数组）中的元素用一个下标数表示，二维矩阵由两个下标数构成，以逗号分开，三维矩阵则由三个下标数构成。如 a(2,3)表示变量 a 的第 2 行第 3 列元素。

在 MATLAB 中，也可以单独给元素赋值，例如，a(2,3)=10，x(1,2)=1.5 等。如果赋值元素的下标超出了原有矩阵的大小，矩阵的行列会自动扩展。

例如，首先输入一变量：

```
>> a=[1 2 3;4 5 6;7 8 9]
```

输出如下：

```
a =
     1     2     3
     4     5     6
     7     8     9
```

再输入：

```
>> a(4,5)=4.4
```

输出如下：

```
>> a(4,4)=4.4
a =
    1.0000    2.0000    3.0000         0
    4.0000    5.0000    6.0000         0
    7.0000    8.0000    9.0000         0
         0         0         0    4.4000
```

可见，变量 a 的阶数由 3×3 自动扩展成 4×4 阶，且元素 a(4,1)、a(4,2)、a(4,3)、a(1,4)、a(2,4)及 a(3,4)被自动赋值为 0。这种自动扩展阶数的功能只适用于赋值语句。在其他语句中若出现超阶调用矩阵元素的情况，MATLAB 将给出出错提示。

变量的阶数可以用 size 命令来获取，例如：

```
>> size(a)
```

输出如下：

```
ans =
     4     4
```

此时，MATLAB 自动给出一个临时变量“ans”。

3．赋值技巧

在 MATLAB 中，为变量的赋值提供一些简单快捷的方法。

1）利用冒号“:”给全行的元素赋值

例如，给 a 的第 5 行全行赋值，可用冒号“:”，输入：

```
>> a(5,:)=[5 3 2 1]   %注意冒号“:”的功能
```

输出如下：

```
a =
    1.0000    2.0000    3.0000         0
    4.0000    5.0000    6.0000         0
    7.0000    8.0000    9.0000         0
         0         0         0    4.4000
    5.0000    3.0000    2.0000    1.0000
```

2）利用行、列标注构成的矩阵

例如，把 a 的第 2 行、第 4 行及第 1 列、第 3 列交点的元素取出，构成一个新矩阵 b，可输入：

```
>> b=a([2,4],[1,3])
```

输出如下：

```
b =
     4     6
     0     0
```

又如要抽去 a 中的第 2 行、第 4 行、第 5 行，可利用空矩阵“[]”的概念，输入：

```
>> a([2,4,5],:)=[]
```

输出如下：

```
a =
     1     2     3     0
     7     8     9     0
```

矩阵的阶数由 5×4 阶降为 2×4 阶。这里值得注意的是，空矩阵与零矩阵是两个不同概念。空矩阵是指没有元素的矩阵，对任何一个矩阵赋值为[]，就是使它的元素都消失掉。零矩阵中元素是存在的，只是其数值为零。因此，利用空矩阵可以缩减矩阵的阶数。

4．特殊矩阵和数组

除了采用直接输入方法对变量赋值外，也可利用 MATLAB 的内部函数来对变量赋值，利用这些函数来创建和生成特殊矩阵或数组。在 MATLAB 中提供了许多生成矩阵的函数命令，这些函数命令存放在“matlabe\elmat”目录下。表 1-2 给出一些常用的生成矩阵函数。利用这些函数，可以直接生成一个矩阵或数组。关于这些函数的具体用法，可以利用 help 命令来获得，下面仅对一些函数命令的使用作简单说明。

表 1-2　常用的生成矩阵函数

函 数 名 称	功 能 描 述
zeros	生成一个元素全部为 0 的矩阵或数组
ones	生成一个元素全部为 1 的矩阵或数组
eye	生成一个单位矩阵或数组
pascal	生成一个帕斯卡矩阵或数组
magic	生成一个魔方矩阵
linspace	生成一个线性间隔的行向量
logspace	生成一个对数间隔的行向量
diag	由矩阵 ***A*** 的主对角线元素得到一个列向量
rand	生成随机矩阵或数组，元素在 (0, 1)之间服从均匀分布
randn	生成随机矩阵或数组，元素服从均值为 0、方差为 1 的正态分布

1）单位矩阵 eye

功能：产生主对角线元素为 1、其他元素为 0 的单位矩阵。其调用格式如下：

```
Y = eye(n)
Y = eye(m,n)
eye([m n])
Y = eye(size(A))
eye(m, n, classname)
eye([m,n],classname)
```

例如：

```
>> x = eye(2,3,'int8')
```

输出如下：

```
x =
     1     0     0
     0     1     0
```

2）零矩阵 zeros

功能：生成一个元素全部为 0 的矩阵或数组，其调用格式如下：

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
B = zeros(m,n,p,...)
B = zeros([m n p ...])
B = zeros(size(A))
zeros(m, n,...,classname)
```

zeros([m,n,...],classname)

其中：ones 函数、rand 函数、randn 函数与 zeros 函数调用格式类似。

例如：

```
>> Z=zeros(3,4)     %产生 3×4 的零矩阵
Z =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> A=ones(3,3,2)    %产生 3×3×2 大小的"1"矩阵
A(:,:,1) =
     1     1     1
     1     1     1
     1     1     1
A(:,:,2) =
     1     1     1
     1     1     1
     1     1     1
>> X=randn(2,5)    %产生均匀为 0,方差为 1 的正态分布的随机一维数组或列向量
X =
    0.5377   -2.2588    0.3188   -0.4336    3.5784
    1.8339    0.8622   -1.3077    0.3426    2.7694
>> Y=rand(5,2)    %产生(0,1)之间均匀分布的随机一维数组或行向量
Y =
    0.1576    0.1419
    0.9706    0.4218
    0.9572    0.9157
    0.4854    0.7922
    0.8003    0.9595
```

3）linspace 函数

功能：linspace 函数将指定区间[a, b]按线性等分。其调用格式如下：

y = linspace(a,b)

y = linspace(a,b,n)

例如：

```
>> Y=linspace(1,2,5)    %把区间[1,2] 5 等分
```

输出如下：

```
Y =
    1.0000    1.2500    1.5000    1.7500    2.0000
```

4）logspace 函数

功能：logspace 函数则是将区间[a, b]按对数等分。其调用格式如下：

y = logspace(a,b)

y = logspace(a,b,n)

y = logspace(a,pi)

例如：

```
>> Y=logspace(1,2,5)    %把区间[1,2]按对数 5 等分
```

输出如下：

```
Y =
    10.0000    17.7828    31.6228    56.2341   100.0000
```

5．复数的赋值方式

MATLAB 的每一个元素都可以是复数，实数是复数的特例。复数的虚数部分用 i 或 j 表示，这是在 MATLAB 启动时就自动设定的。例如，输入：

```
>> c=3-5.3i
```

输出如下：

```
c =
    3.0000 - 5.3000i
```

对复数矩阵有两种赋值方法。

方法 1：可将矩阵元逐个赋予复数。例如，输入：

```
>> z=[1+2i,3-4i;5+2i,5-2i]
```

输出如下：

```
z =
    1.0000 + 2.0000i    3.0000 - 4.0000i
    5.0000 + 2.0000i    5.0000 - 2.0000i
```

方法 2：将矩阵的实数和虚部分别赋值。例如，输入：

```
>> z=[1,2;5,5]+[2,-4;2,-2]*i
```

输出如下：

```
z =
    1.0000 + 2.0000i    2.0000 - 4.0000i
    5.0000 + 2.0000i    5.0000 - 2.0000i
```

由结果可知，两种方法可得出同样结果。但值得注意的是：

（1）在方法 2 中省略乘号“*”，就会出错。例如：

```
>> z=[1,2;5,5]+[2,-4;2,-2]i
??? z=[1,2;5,5]+[2,-4;2,-2]i
                           |
Error: Unexpected MATLAB expression.
```

（2）如果在前面其他程序中曾经给 i 或 j 赋过值，这时就不能采用方法 1 或方法 2 乘号“*”方式对复数赋值，但仍可采用方法 1 中非乘号“*”方式对复数赋值。这是因为 i 和 j 已经不是虚数符号。例如，若事先已赋值 i=5，这时再输入：

```
>> z=[1,2;5,5]+[2,-4;2,-2]*i
```

输出如下：

```
z =
    11    -18
    15     -5
```

若要采用乘号“*”方式对复数赋值，此时应输入：

```
>> clear i j
```

6．MATLAB 内部特殊变量和常数

在 MATLAB 内部，为处理方便定义了一些特殊变量和常数。其介绍分别如下：

（1）ans：临时变量，通常指示当前的答案。

（2）eps：常数，表示浮点相对精度；其值是从 1.0 到下一个最大浮点数之间的差值。按 IEEE 标准，eps=2^{-52}，近似为 2. 2204e−016，该变量值作为 MATLAB 一些函数计算的相对浮点精度。

（3）realmax：常数，表示最大正浮点数，任何大于该值的运算都溢出。在具有 IEEE 标准浮点格式的机器上，realmax 略小于 2^{1024}，近似为 1.7977e+308。

（4）realmin：常数，表示最小正浮点数，任何小于该值的运算都溢出。在具有 IEEE 标准浮点格式的机器上，realmax 略小于 2^{-1024}，近似为 2.2251e−308。

（5）pi：常数，表示圆周率π=3.1415926535897…。表达式 4*atan(1)和 imag(log(−1))产生相同的值π。

（6）Inf：常数，代表正无穷大；一般被 0 除或溢出则产生无穷大结果。如 2/0、2^10000 均产生结果 Inf；而 log(0)产生结果−Inf。

（7）i, j：虚数单位，表示复数虚部单位，相当于 $\sqrt{-1}$ 。

（8）NaN：表示非数值。如 Inf-Inf、Inf/Inf、0*Inf、0/0 均产生该结果。

7．变量检查

在程序调试或变量的赋值过程中，往往需要检查工作空间中的变量、变量的阶数及变量赋值内容。在检查变量及其阶数等内容时，既可用工作空间窗口，也可在命令窗口使用 who 命令或 whos 命令来完成检查。当查看某变量的赋值情况时，可在命令窗口直接输入该变量名并回车即可。若所查变量不存在（如变量 yy），MATLAB 将给出如下提示信息：

```
>> yy
??? Undefined function or variable 'yy'.
```

表示目前没有定义 yy 变量或函数。

1.4　MATLAB 的矩阵运算

MATLAB 中的大多数运算可以直接对矩阵应用。除了算术运算+、−、*、^、/和\外，还有用于转置和共轭的运算符、有理数运算符和逻辑运算符。此外，矩阵有算术函数和逻辑函数，有些函数仅能在二维矩阵中使用。

1.4.1　矩阵的加减法

如果矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 具有相同的维数，那么就可以定义两个矩阵的和 $\boldsymbol{A}+\boldsymbol{B}$ 和两个矩阵的差 $\boldsymbol{A}-\boldsymbol{B}$，即矩阵 $\boldsymbol{A}\pm\boldsymbol{B}$，元素 $a_{ij}\pm b_{ij}$。在 MATLAB 中，一个 $m\times n$ 矩阵 $\boldsymbol{A}$ 和一个标量，即一个 1×1 矩阵 $\boldsymbol{S}$ 之间也能进行加减运算。矩阵 $\boldsymbol{A}+\boldsymbol{S}$ 得到与 $\boldsymbol{A}$ 相同的维数，元素为 $a_{ij}+s$。

【例 1-2】矩阵的加减法运算。

```
>> a=[1 7;2 8];
b=[5 0;4 3];
add=a+b
sub=a-b
add99=a+99
sub88=a-88
```

运行程序，输出如下：

```
add =
     6     7
     6    11
sub =
    -4     7
```

```
      -2       5
add99 =
    100     106
    101     107
sub88 =
    -87     -81
    -86     -80
```

1.4.2 矩阵的乘除

1．矩阵的乘法运算

如果矩阵 ***A*** 的列数等于矩阵 ***B*** 的行数，那么矩阵相乘，即 ***C*=*AB***，就被定义为二维矩阵。如果不是这种情况，MATLAB 就返回一个错误信息。只有一个例外，就是这两个矩阵之一是 1×1。如果一个标量，那么 MATLAB 是可以接受的。在 MATLAB 中，乘法的运算符是*，因此，命令是 ***C*=*A*∗*B***。

元素 c_{ij} 是 ***A*** 的第 *i* 行和 ***B*** 的第 *j* 列的点积。矩阵 ***C*** 有与 ***A*** 相同的行数和 ***B*** 相同的列数。对于方阵，也定义了积 ***BA***，但其结果通常与 ***AB*** 不同。

【例 1-3】矩阵的乘法运算。

```
>> A=[1 0 ;0 1];
B=[0 1 ;1 0];
multAB=A*B,
multBA=B*A
```

运行程序，输出如下：

```
multAB =
     0     1
     1     0
multBA =
     0     1
     1     0
```

可以看到 A∗B 和 B∗A 的结果相同。

```
>> A=[1 2;3 4];
B=[6 5;7 8];
multAB=A*B
multBA=B*A
```

运行程序，输出如下：

```
multAB =
    20    21
    46    47
multBA =
    21    32
    31    46
```

可以看到 A∗B 和 B∗A 的结果不同。

```
>> a=[1 4 7];
b=[1;10;100];
S = a*b
M = b*a
```

运行程序，输出如下：

```
S =     741
M =
       1       4       7
      10      40      70
     100     400     700
```

可以看到，左右相乘产生不同维数的矩阵情况。

2．矩阵的除法运算

在 MATLAB 中，矩阵除法有两种运算符“\”和“/”，分别表示矩阵运算左除和右除。如果 $\boldsymbol{A}$ 或 $\boldsymbol{B}$ 为非奇异矩阵，则 $\boldsymbol{A}\backslash\boldsymbol{B}=\boldsymbol{A}^{-1}*\boldsymbol{B}$ 或 $\boldsymbol{A}/\boldsymbol{B}=\boldsymbol{A}*\boldsymbol{B}^{-1}$。其中，$\boldsymbol{A}^{-1}$ 是矩阵 $\boldsymbol{A}$ 的逆。MATLAB 提供了用于求矩阵的逆矩阵的函数 inv，因此可以用 inv(A)求矩阵 $\boldsymbol{A}$ 的逆矩阵。一般情况下，$x=a\backslash b$ 是方程 $a*x=b$ 的解，而 $x=b/a$ 是方程 $x*a=b$ 的解，两种运算的结果不相等。

与矩阵的除法运算相类似，数组也具有除法运算符。运算符“.\”和“./”分别表示数组运算的左除和右除，表示数组相应元素相除。进行数组除法运算时两数组必须大小相同，除非其中有一个是标量。

【例 1-4】矩阵的除法运算。

```
>> A=[1 0 ;0 1];
B=[0 1 ;1 0];
invAB=A\B
invBA=A/B
```

运行程序，输出如下：

```
invAB =
     0     1
     1     0
invBA =
     0     1
     1     0
```

可以看出，矩阵的左除等于矩阵的右除。

```
>> A=[1 2;3 4];
B=[6 5;7 8];
invAB=A\B
invBA=A/B
```

运行程序，输出如下：

```
invAB =
   -5.0000   -2.0000
    5.5000    3.5000
invBA =
   -0.4615    0.5385
   -0.3077    0.6923
```

可以看出，矩阵的左除不等于矩阵的右除。

1.4.3　MATLAB 索引或引用

在 MATLAB 中有 3 种基本方法可以选取一个矩阵的子阵。它们分别是下标法、线性法和逻辑法。

1．下标法

矩阵中的元素是按照行列规律排列的，因此元素的调用可以用其行号加列号进行调用。一般对矩阵元素的行列号表示使用下标表示，例如，$\boldsymbol{A}_{1,3}$，$\boldsymbol{A}_{1,5}$ 表示矩阵 $\boldsymbol{A}$ 中位于第 1 行、第 3 列和第 5 列的元素，行数为 1 时可以省略。$\boldsymbol{A}_{2,3}$ 表示矩阵中位于第 2 行、第 3 列的元素。

【例 1-5】下标法。

```
>> A = 3:9
A =
     3     4     5     6     7     8     9
>> A([4,7])          %取第 4 个和第 7 个元素
ans =
     6     9
>> A([4:1:end])      %从第 4 个元素开始,每隔 1 个元素取一个元素,直到最后
ans =
     6     7     8     9
>> B=[11 12 13;14 15 16;17 18 19]
B =
    11    12    13
    14    15    16
    17    18    19
>> B(2:3,2)          %取第 2 行和第 3 行的第 2 列元素
ans =
    15
    18
```

2．线性法

二维矩阵以列优先顺序线性展开，可以通过线性展开后的元素序号来访问元素。例如，要取矩阵 $\boldsymbol{A}$ 的第 3 行、第 2 列的元素，下标法是 $\boldsymbol{A}_{3,2}$，而由列向顺序来排，该元素位于第 6 位，因此，可以把 $\boldsymbol{A}$ 当成是行数为 1 的矩阵，直接使用序号 6 调取该元素。

【例 1-6】线性法。

```
>> A = [11 14 17; 12 15 18; 13 16 19]
A =
    11    14    17
    12    15    18
    13    16    19
>> A(5)              %取列向排列第 5 个元素
ans =
    15
>> A([2,3,8])        %取列向排列第 3,第 1 和第 8 个元素,取出元素按照单行排序
ans =
    12    13    18
>> A([3;1;8])        %取出元素按照单列排序
ans =
    13
    11
    18
```

3．逻辑法

用一个和原矩阵具有尺寸的 0-1 矩阵，可以索引元素。在某个位置上为 1 表示选取元素，否则不选。得到的结果是一个向量。由于 2 个矩阵的尺寸相同，逻辑矩阵可以看成是一个图像遮罩，如果是 1，原矩阵中的元素可以透过去，否则就被逻辑遮罩给挡住了，透不过去。

【例 1-7】逻辑法。

```
>> A = 6:10;
>> B=5:10
B =
     5     6     7     8     9    10
%矩阵 A 中对应逻辑矩阵中数值为 1 的元素可以保留,其余被过滤掉
>> B(logical([0 0 1 0 1]))
ans =
     7     9
>> A=[1 2 3 4];
>> B=[1 0 0 1];
>> B(logical(A))
ans =
     1     0     0     1
>> A(logical(B))
ans =
     1     4
```

1.4.4　数组操作和矩阵操作

对矩阵的元素一个一个孤立进行的操作称做数组操作；而把矩阵视为一个整体进行的运算则称为矩阵操作。MATLAB 运算符*、/、\、^都是矩阵运算，而相应的数组操作则是.*、./、.\、.^。

【例 1-8】数组操作和矩阵操作。

```
>> A=[1 0 ;0 1];
B=[0 1 ;1 0];
M=A*B
S=A.*B
```

运行程序，输出如下：

```
M =                    %矩阵相乘
     0     1
     1     0
S =                    %数组相乘
     0     0
     0     0
```

1.4.5　布尔数组操作

对矩阵的比较运算是数组操作，也就是说，是对每个元素孤立进行的。因此其结果就不是一个“真”或者“假”，而是一维“真假”。这个结果就是布尔数组。

【例 1-9】布尔数组操作。

```
>> A=[-1 0 0.2 -2 -2.2 1.5 1.6 7];
>> A>1
```

```
ans =
     0     0     0     0     0     1     1     1
```

如果想选出 A 中的负元素：

```
>> A=A(A<0)
A =
   -1.0000    -2.0000    -2.2000
```

除此之外，MATLAB 运算中出现 NaN、Inf、-Inf，通过下列程序来了解它们：

```
>> Inf==Inf
ans =
     1
```

同时，可以用 isinf、isnan 判断。

1．isinf 函数

其调用格式如下：

TF = isinf(A)：返回与 A 尺寸相同的数组，如果包含逻辑数 1（真），表示对应 A 中元素是正无穷大或者负无穷大；如果包含逻辑数 0（假），则表示对应 A 中元素不是无穷大。对于一个复数 Z，无论复数的实部还是虚部为无穷大，isinf(z)都返回 1；如果实部和虚部为有限值或空，isinf(z)返回。

对于任意实数 A，isfinite(A)、isinf(A)和 isnan(A)返回的数值只有一个为 1。

【例 1-10】isinf 函数。

```
>> a = [-2   -1   0   1   2]
a =
    -2     -1      0      1      2
isinf(1./a)
ans =
     0     0     1     0     0
>> isinf(0./a)
ans =
     0     0     0     0     0
```

2．isnan 函数

其调用格式如下：

TF = isnan(A)：返回与 A 尺寸相同的数组，如果包含逻辑数 1（真），表示对应 A 中元素为空；如果包含逻辑数 0（假），则表示对应 A 中元素不为空。对于一个复数 z，无论复数的实部还是虚部为空，isinf(z)都返回 1；如果实部和虚部都为有限小或无穷大，isinf(z)返回 0。

对于任意实数 A，isfinite(A)、isinf(A)和 isnan(A)返回的数值只有一个为 1。

【例 1-11】isnan 函数。

```
>> a = [-2   -1   0   1   2]
isnan(1./a)
isnan(0./a)
```

运行程序，输出如下：

```
a =
    -2     -1      0      1      2
ans =
     0     0     0     0     0
ans =
```

```
     0     0     1     0     0
```

比较两个矩阵大小时，矩阵必须具有相同的尺寸，否则会出错。使用 size 函数可以知道矩阵的大小尺寸，使用 isequal 函数可以比较数组是否相等。

1.5　M 文件的类型

本节将要对 M 文件的基本类型展开介绍，主要包括数据文件、函数式 M 文件及脚本式 M 文件。

1.5.1　数据文件

数据文件是 MATLAB 中经常使用的用于保存变量的文件，该文件的后缀为 mat。数据文件是以标准二进制格式将变量进行保存的一种格式，使用数据文件可将工作空间中的全部或部分数据变量保存下来。数据文件的生成和调用是由函数 save 和 load 完成的。

【例 1-12】数据文件的调用。

在 MATLAB 的命令窗口中输入以下代码：

```
>> load woman     %woman 是 MATLAB 自带的一个图像数据文件
>> image(X)       %显示该图像文件
```

运行程序，效果如图 1-9 所示。

更改图像的色图，并刷新图形显示。其代码如下：

```
>> colormap(gray)
```

效果如图 1-10 所示。

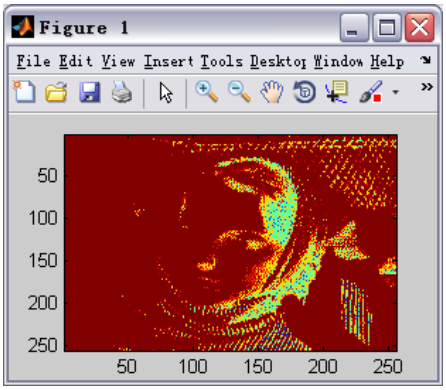


图 1-9　mat 文件的调用效果 A

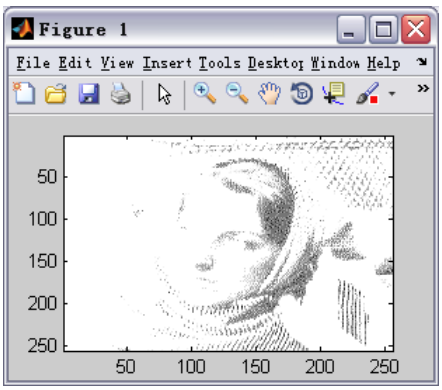


图 1-10　mat 文件的调用效果 B

1.5.2　M 文件

M 文件的编程语法类似于 C 语言，但又有其自身特点。M 文件只是一个简单的 ASCII 码文本文件，执行程序时逐行解释运行程序。M 文件可分为两种类型：命令脚本文件和函数文件，其中函数文件又可分为内置函数、自定义函数、S 函数等各种类型。

1．命令脚本文件

命令脚本文件实际上是一串指令的集合。命令脚本文件的执行结果与在命令窗口逐行执行文件中的所有指令的结果是一样的。命令脚本文件没有输入、输出参数。文件运行后，产生的

所有变量都保存在 MATLAB 的基本工作空间中，除非用户使用 clear 指令清除或关闭 MATLAB，否则这些变量将一直保存在基本工作空间中。

命令脚本文件包括两部分：注释部分和程序部分。其中注释部分必须在符号“%”之后，MATLAB 不对其进行计算，只供程序设计人员和阅读者方便理解程序而用。程序部分就是程序中一般的命令行和程序段，MATLAB 要对其进行编译和计算。

2．MATLAB 内置函数文件

MATLAB 自定义的函数文件称内置函数文件。调用内置函数的方法是使用函数名并给出相应的入口、出口参数即可。

【例 1-13】MATLAB 的内置函数使用。

其实现的 MATLAB 程序代码如下：

```
>> x=0:pi/10:pi;          %生成变量 x，下面调用内置函数 cos 和绘图函数 plot
>> y=sin(x);
>> plot(x,y)
```

运行程序，效果如图 1-11 所示。

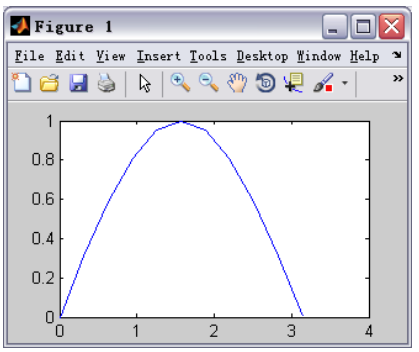


图 1-11　MATLAB 内置函数的调用

3．用户自定义的 M 函数文件

MATLAB 用户可以根据需要编辑自己的 M 函数文件，它们可以像 MATLAB 提供的库函数一样被方便地调用。这种用 MATLAB 语言创建与定义新函数的功能，体现了 MATLAB 语言强大的扩展功能。

用户自定义函数文件编写时需要有输入变量和输出变量。M 函数文件一般格式如下：

```
function    返回变量=函数名（输入变量）
% 注释说明语句段
程序语句段
```

M 函数文件第一行必须以关键字 function 作为引导词，M 函数文件的文件名必须是“<函数名>.m”。程序中的变量均为局部变量，不保存在工作空间中，变量只在函数运行期间有效。

【例 1-14】应用用户自定义函数来求解一元二次方程 $ax^2+bx+c=0$ 的根。

打开 M 文件编辑器，输入以下内容，并保存为 li1_14.m 文件。

```
%function li1_14  用于求解一元二次方程的根
function [x1,x2]=li1_14(a,b,c)
x1=(-b+sqrt(b*b-4*a*c))/(2*a);
x2=(-b-sqrt(b*b-4*a*c))/(2*a);
```

当 a=1、b=2、c=0 时，在命令窗口直接调用自定义函数 li1_14 来求解方程的根。

```
>> [x1,x2]=li1_14(1,2,0)
```

求解方程的根如下：

```
x1 =
     0
x2 =
    -2
```

4．系统文件 S 函数

系统文件 S 函数用于描述系统运动的专用函数，是特殊的 M 文件。创建 S 函数的方法有 3 种：一是由 Simulink 结构图自动创建；二是用函数 M 文件编写；三是采用 C 语言编写 mex 文

件直接定义。S 函数一旦创建，即可在框图中使用，也可在文件中调用。S 函数的一般调用格式如下：

[sys, x0]=sfunction (t, x, u, flag)

其中：sys 是系统状态，x0 是状态初值，sfunction 表示用户定义的系统，t 是当前时刻；x 是当前状态，u 是当前输入值，flag 是标志量。

S 函数与函数 M 文件类似，只是输入、输出变量是限定的。

5．函数句柄

函数句柄是 MATLAB 6 之后特有的语言结构，优点是方便地实现函数间互相调用，兼容函数加载的所有方式、拓宽子函数包括局部函数的使用范围、提高函数调用的可靠性；减少程序设计中的冗余、提高重复执行的效率、数组、结构数组、细胞型数组能够结合定义数据。

定义函数句柄只需在提示符@后添加相应函数的函数名，函数句柄的内容通过 functions 显示。调用可通过函数 feval 进行，调用格式如下：

feval(函数句柄，参数列表)

函数句柄与函数名字符串转换可通过以下代码实现：

```
函数句柄名=str2func('函数名字符串')     %函数名字符串转换函数句柄
func2str(函数句柄名)                    %函数句柄转换函数名字符串
```

【例 1-15】函数句柄的创建和显示。

```
>> f_h=@sym
```

创建的函数句柄如下：

```
f_h =
    @sym
```

在命令窗口中输入以下代码：

```
>> functions(f_h)
```

返回的函数句柄如下：

```
ans =
    function: 'sym'
        type: 'simple'
        file: ''
```

【例 1-16】函数句柄的调用。

首先在命令窗口中输入以下代码，创建函数句柄 f1。

```
>> f1=@rand
f1 =
    @rand
>> functions(f1)
```

返回的函数句柄如下：

```
ans =
    function: 'rand'
        type: 'simple'
        file: 'MATLAB built-in function'
>> feval(f1,6)
ans =
    0.8909    0.8407    0.1966    0.5853    0.3804    0.9340
```

```
0.9593    0.2543    0.2511    0.5497    0.5678    0.1299
0.5472    0.8143    0.6160    0.9172    0.0759    0.5688
0.1386    0.2435    0.4733    0.2858    0.0540    0.4694
0.1493    0.9293    0.3517    0.7572    0.5308    0.0119
0.2575    0.3500    0.8308    0.7537    0.7792    0.3371
```

1.6 MATLAB 程序结构流

程序的结构有顺序结构、选择结构和循环结构 3 种。任何复杂的程序都是由这 3 种基本结构所构成的。

1.6.1 顺序结构流

顺序结构是指按照程序中语句的排列顺序次序执行，直到程序的最后一个语句。这是最简单的一种程序结构。一般涉及数据的输入、数据的计算或处理、数据的输出等内容。

1．数据的输入

从键盘上输入数据，可以使用 input 函数，其调用格式如下：

a=input（提示信息，选项）：其中，提示信息为一个字符串，用于提示用户输入什么样的数据。

例如，从键盘输出正整数 n，可以采用以下命令来完成。

```
n=input('输入正整数 n=');
```

执行该语句时，首先在屏幕上显示提示信息“输入正整数 n=”，然后等待用户从键盘上输入正整数 n 的值。

如果在 input 函数调用时采用's'选项，则允许用户输入一个字符串。例如，想输入一个人的姓名，可采用命令：

```
name=input('请输入姓名', 's')
```

2．数据的输出

MATLAB 提供的命令窗口输出函数主要有 disp，fprintf 函数。disp 函数调用格式如下：

disp(输出项)：输出项可以是字符串，也可以是矩阵。例如：

```
>> A='您好';
>> disp(A)
```

输出如下：

```
您好
```

又如：

```
>> A=[1 4 7;2 5 8;3 6 9];
>> disp(A)
```

输出如下：

```
     1     4     7
     2     5     8
     3     6     9
```

fprintf 函数最常见的使用方式用以下例子来说明。

若输入命令：

```
fprintf('圆周率 pi=%10.9f',pi)
```

则会按浮点型数输出含 9 位小数，1 位整数的圆周率近似值，其输出结果如下：

```
圆周率 pi=3.141592654
```

若输入命令：

```
>> n=24; fprintf('n=%d',n)
```

则会按整型数输出 n 的值，其输出结果为如下：

```
n=24
```

若输入命令：

```
>> n=24;fprintf('n=%f',n)
```

则会按浮点型数输出 n 的值，其输出结果如下：

```
n=24.000000
```

3．程序的暂停

当程序运行时，为了查看程序的中间结果或者观看输出的图形，有时需要暂停程序的执行，这时可以使用 pause 函数，其调用格式如下：

```
pause(延迟秒数)
```

如是省略延迟秒数，则将暂停程序，直到用户按任一键后程序继续执行。

1.6.2　选择结构流

选择结构是根据给定的条件成立或不成立，分别执行不同的语句。MATLAB 用于实现选择结构的常用语句有 if 语句和 switch 语句。

1．if 语句

在 MATLAB 中，if 语句有 3 种格式。

1）单分支 if 语句

语句格式如下：

```
if  条件
      语句组
end
```

当条件成立时，则执行语句组，执行完之后继续执行 if 语句的后续语句，若条件不成立，则直接执行 if 语句的后继语句。例如，当 x 是整数矩阵时，输出 x 的值，语句如下：

```
if fix(x)==x
     disp(x);
end
```

2）双分支 if 语句

语句格式：

```
if  条件
      语句组 1
else
      语句组 2
end
```

当条件成立时，执行语句组 1，否则执行语句组 2，语句组 1 或语句组 2 执行后，再执行 if 语句的后继语句。

3）多分支 if 语句

语句格式如下：

```
if  条件 1
      语句组 1
elseif  条件 2
      语句组 2
⋮
elseif  条件 m
      语句组 m
else
      语句组 n
end
```

下面通过一个示例来说明 if 语句的用法。

【例 1-17】生成一个矩阵。

```
for m=1:k
    for n=1:k
        if m= =n
            a(m,n)=2;
        elseif abs(m-n)= =2
            a(m,n)=1;
        else
            a(m,n)=0;
        end
    end
end
```

当 k=6 时，输出一个矩阵如下：

```
a =
     2     0     1     0     0     0
     0     2     0     1     0     0
     1     0     2     0     1     0
     0     1     0     2     0     1
     0     0     1     0     2     0
     0     0     0     1     0     2
```

2．switch 语句

switch 语句称为条件选择语句，其关键字包括 switch、case、otherwise 和 end。它主要用于有选择性的程序设计，实现程序的多分支选择。语句格式如下：

```
switch   选择表达式
    case   情况表达式 1
          语句组 1
    case  情况表达式 2
          语句组 2
```

```
        ⋮
        otherwise
        语句组 n
    end
```

其工作流程是，计算机首先计算选择表达式的值，然后将该值分别与情况表达式的值进行比较；如果与其中之一相等，则执行该情况表达式后的语句组，执行完该语句组后跳至 end，执行 end 后续语句；如果与任何一个情况表达式的值都不等，则执行 otherwise 后的语句组 n。

switch 子句后面的表达式应为一个标量或一个字符串，case 子句后面的表达式不仅可以为一个标量或一个字符串，而且还可以为一个单元矩阵。如果 case 子句后面的表达式为一个单元矩阵，则表达式的值等于该单元矩阵中的某个元素时，执行相应的语句组。

【例 1-18】学生的成绩管理作为演示 switch 结构的应用，划分区域：满分（100）、优秀（90～99）、良好（80～89）、及格（60～79）、不及格（<60）。

其实现的 MATLAB 程序如下：

```
>> for i=1:10
       a{i}=89+i;
       b{i}=79+i;
       c{i}=69+i;
       d{i}=59+i;
   end
   c=[d,c];
   Name={'zhang','Li','huang','chen','zhu'};   %元胞数组
   Score={82,91,89,40,100};
   Rank=cell(1,5);
```

创建一个含有 5 个元素的结构体数组 S，它有 3 个域：Name、Score、Rank：

```
>> S=struct('Name',Name,'Score',Score,'Rank',Rank);
```

根据学生的分数，求出相应的等级：

```
>> for i=1:5
       switch S(i).Score
           case 100
               S(i).Rank='满分';
           case a
               S(i).Rank='优秀';
           case b
               S(i).Rank='良好';
           case c
               S(i).Rank='及格';
           otherwise
               S(i).Rank='不及格';
       end
   end
```

将学生的姓名、得分、登记等信息打印出来：

```
>> disp(['学生姓名     ','得分     ','等级']);
for i=1:5
```

```
    disp([S(i).Name,blanks(6),num2str(S(i).Score),blanks(6),S(i).Rank]);
end
学生姓名      得分      等级
zhang         82        良好
Li            91        优秀
huang         89        良好
chen          40        不及格
zhu          100        满分
```

1.6.3 循环结构流

循环是指按照给定的条件，重复执行指定的语句，这是一种十分重要的程序结构。MATLAB 用于实现循环结构的语句有 for 语句、while 语句。

1．for 语句

for 语句的格式如下：

```
for 循环变量=表达式 1：表达式 2：表达式 3
    循环体语句
end
```

其中：表达式 1 的值为循环变量的初值，表达式 2 的值为步长，表达式 3 的值为循环变量的终值。步长为 1 时，表达式 2 可以省略。

for 语句执行过程为：首先计算 3 个表达式的值，再将表达式 1 的值赋给循环变量，如果此时循环变量的值介于表达式 1 和表达式 3 的值之间，则执行循环体语句，否则结束循环的执行。执行完一次循环之后，循环变量自增一个表达式 2 的值，然后再判断循环变量的值是否介于表达式 1 和表达式 3 之间，如果满足，仍然执行循环体，直到不满足为止。这时将结束 for 语句的执行，而继续执行 for 语句的后继语句。

【例 1-19】设 $f(x)=\mathrm{e}^{-0.5x}\sin\left(x+\dfrac{\pi}{6}\right)$，求 $s=\int_0^{3\pi} f(x)\mathrm{d}x$。

求函数 $f(x)$在$[a,b]$上的定积分，其几何意义就是求曲线 $y=f(x)$与直线 $x=a$、$x=b$、$y=0$ 所围成的曲边梯形的面积。为了求得曲边梯形面积，先将积分区间$[a,b]$分成 n 等份，每个区间的宽度为 $h=(b-a)/n$，对应地将曲边梯形分成 n 等份，每个小部分即是一个小曲边梯形。近似求出每个小曲边梯形面积，然后将 n 个小曲边梯形的面积加起来，就得到总面积，即定积分的近似值。近似地求每个小曲边梯形的面积，常用的方法有矩形法、梯形法以及辛普森法则等。以梯形法为例，程序代码如下：

```
>> clear all;
a=0;   b=3*pi;
n=1000; h=(b-a)/n;
x=a;   s=0;
f0=exp(-0.5*x)*sin(x+pi/6);
for i=1:n
    x=x+h;
    f1=exp(-0.5*x)*sin(x+pi/6);
    s=s+(f0+f1)*h/2;
    f0=f1;
end
```

```
s
```

运行程序，输出如下：

```
s =
    0.9008
```

上述程序来源于传统的编程思想。也可以利用向量运算，从而使得程序更加简洁，更有 MATLAB 的特点。其源程序如下：

```
>> clear all;
a=0;   b=3*pi;
n=1000; h=(b-a)/n;
x=a:h:b;
f=exp(-0.5*x).*sin(x+pi/6);
for i=1:n
    s(i)=(f(i)+f(i+1))*h/2;
end
s=sum(s)
s =
    0.9008
```

程序中 x、f、s 均为向量，f 的元素为各个 x 点的函数值，s 的元素分别为 n 个梯形的面积，s 各元素之和即定积分近似值。

2．while 语句

while 语句的一般格式如下：

```
while   (条件)
        循环体语句
end
```

while 语句的执行过程：若条件成立，则执行循环体语句，执行后再判断条件是否成立，如果成立，则继续执行循环体语句，如果不成立则跳出循环。

【例 1-20】从键盘输入若干个数，当输入 0 时结束输入，求这些数的平均值和它们的和。

其实现的程序代码如下：

```
>> clear all;
sum=0; n=0;
x=input('Enter a number (end in 0):');
while (x~=0)
    sum=sum+x;
    n=n+1;
    x=input('Enter a number (end in 0):');
end
if (n>0)
    sum
    mean=sum/n
end
```

运行程序输出结果如下：

```
Enter a number (end in 0):89
Enter a number (end in 0):37
Enter a number (end in 0):25
Enter a number (end in 0):75
```

```
Enter a number (end in 0):196
Enter a number (end in 0):0
sum =
    422
mean =
    84.4000
```

1.7 MATLAB 的数据结构

在 MATLAB 中提供了许多内置函数，下面对一些常用的内置函数作简单介绍。

1．max 函数

功能：求最大值。其调用格式如下：

C = max(A)：沿数组不同的维，返回最大值元素。如果 A 是向量，返回 A 中最大值元素；如果 A 是矩阵，则处理矩阵列作为向量，返回一个行向量，其元素为矩阵每列中的最大元素；如果 A 为多维数组，则沿第一个非单元素维进行处理，求得各向量的最大值。

C = max(A,B)：返回一个与 A 和 B 一样大小的数组。其元素取 A 或 B 中最大的一个。

C = max(A,[],dim)：返回数组（矩阵）A 由标量 dim 所指定的维数（或行）的最大值。

[C,I] = max(...)：返回最大值的同时，返回一个下标向量。

min 函数为返回最小值，其用法与 max 类似，在此不展开介绍。

【例 1-21】返回最大值、最小值。

```
>> X=[6 9 3 4;11 8 9 2;9 12 13 14;15 0 8 7];
>> C=max(X)     %返回最大值
C =
     15     12     13     14
>> C=min(X)   %返回最小值
C =
      6      0      3      2
>> [M,I]=max(X)
M =
     15     12     13     14
I =
      4      3      3      3
>> [M,I]=min(X)
M =
      6      0      3      2
I =
      1      4      1      2
```

2．mean 函数

功能：求平均值。其调用格式如下：

M = mean(A)：沿数组不同的维，返回元素平均值。如果 A 为向量，则返回向量 A 的平均值；如果 A 为矩阵，则将矩阵每列当作向量处理，返回一个平均值行向量；如果 A 为多维数组，则沿第一个非单元维进行处理，返回一个平均值数组。

M = mean(A,dim)：沿标量 dim 指定的维计算元素平均值。

medain 函数为求中间值，其用法与 mean 函数用法类似。

【例 1-22】函数求平均值。

```
>> A = [1 2 3; 3 3 6; 4 6 8; 4 7 7];
>> mean(A)                    %求函数的平均值
ans =
    3.0000    4.5000    6.0000
>> mean(A,3)
ans =
     1     2     3
     3     3     6
     4     6     8
     4     7     7
>> median(A)                  %求函数的中间值
ans =
    3.5000    4.5000    6.5000
>> median(A,3)
ans =
     1     2     3
     3     3     6
     4     6     8
     4     7     7
```

3．sum 函数

功能：求元素的和。其调用格式如下：

B = sum(A)：沿数组不同的维，计算元素和。如果 A 为向量，则返回向量 A 的元素和；如果 A 为矩阵，则将矩阵每列当作向量处理，返回一个元素分别为各列和的行向量；如果 X 为多维数组，则沿第一个非单位元素维进行计算，返回一个元素和数组。

B = sum(A,dim)：沿标量 dim 指定的维数计算元素和。

B = sum(..., 'double')或 B = sum(..., dim,'double')：返回的元素为 double 类型。

B = sum(..., 'native')或 B = sum(..., dim,'native')：返回 A 元素默认的单精度或双精度类型。

【例 1-23】求元素的和。

```
>> M = magic(3)
M =
     8     1     6
     3     5     7
     4     9     2
>> B=sum(M)
B =
    15    15    15
>> B=sum(M,2)
B =
    15
    15
    15
```

4．std 函数

功能：求标准偏差。其调用格式如下：

s = std(X)：X 为向量，则返回 $s=\left[\frac{1}{N-1}\sum_{k=1}^{N}(x_i-\overline{x})^2\right]^{\frac{1}{2}}$ 计算的标准偏差。如果 X 是服从正态分布的随机样本，则 s^2 为其方差的最佳无偏估计；如果 X 为矩阵，则返回矩阵每列标准差的行向量；如果 X 为多维数组，则沿 X 第一非单元素维计算元素的标准差。

s = std(X,flag)：如果 flag=0，与 s=std(X)一样；如果 flag=1，则返回 $s=\left[\frac{1}{N}\sum_{k=1}^{N}(x_i-\overline{x})^2\right]^{\frac{1}{2}}$ 计算的标准方差。

s = std(X,flag,dim)：沿标量 dim 给定的维计算标准差。

【例 1-24】求标准偏差。

```
>> A = [1 2 3; 3 3 6; 4 6 8; 4 7 7];
>> s=std(A)
s =
    1.4142    2.3805    2.1602
>> s=std(A,1)
s =
    1.2247    2.0616    1.8708
>> s=std(A,0,2)
s =
    1.0000
    1.7321
    2.0000
    1.7321
>> s=std(A,1,2)
s =
    0.8165
    1.4142
    1.6330
    1.4142
```

5．cov 函数

功能：求协方差。其调用格式如下：

cov(x)

cov(x) or cov(x,y)

cov(x,1) or cov(x,y,1)

corrcoef 函数为求相关系数，其用法与 cov 函数类似。

【例 1-25】求协分差

```
>> A = [-1 1 2 ; -2 3 1 ; 4 0 3]
A =
    -1     1     2
    -2     3     1
     4     0     3
>> v = cov(A)
v =
```

```
    10.3333   -4.1667    3.0000
    -4.1667    2.3333   -1.5000
     3.0000   -1.5000    1.0000
>> v = cov(A,1)
v =
     6.8889   -2.7778    2.0000
    -2.7778    1.5556   -1.0000
     2.0000   -1.0000    0.6667
>> v = corrcoef(A)
v =
     1.0000   -0.8486    0.9333
    -0.8486    1.0000   -0.9820
     0.9333   -0.9820    1.0000
```

6．diff 函数

功能：求元素之差和近似导数。其调用格式如下：

Y = diff(X)：计算 X 中相邻元素之间的差值。如果 X 为向量，则返回一个比 X 少一个元素的向量，其元素值为[X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)]；如果 X 为矩阵，则返回一个列差值的矩阵：[X(2:m,:)-X(1:m-1,:)]。

Y = diff(X,n)：使用 diff 函数递归 n 次，计算第 n 阶差值，diff(x, 2)=diff(diff(X))。

Y = diff(X,n,dim)：沿 dim 指定的维数计算第 n 阶差值，如果 n 大于或等于 dim 维的长度，则返回空数组。

【例 1-26】求元素之差和近似导数。

```
>> x = [1 2 3 4 5];
y = diff(x)
y =
     1     1     1     1
>> z = diff(x,2)
z =
     0     0     0
```

7．sort 函数

功能：排序。其调用格式如下：

B = sort(A)：沿数组的不同维，以升序排列元素。元素可以为实数、复数和字符串。

如果 X 是一个复数，元素按其模的大小进行排列，如果模相等，则按其在区间[−π,π]上的相角进行排序。

B = sort(A,dim)：沿标量 dim 指定的维对元素排序。如果 dim 是一个向量，则在指定的维进行递归排序。

B = sort(...,mode)：排序指定元素的方向，这取决于模式值。

[B,IX] = sort(A,...)：同时返回一个下标数组 IX。

【例 1-27】元素排序。

```
>> v = [1 -1 i -i];
>> angle(v)
ans =
         0    3.1416    1.5708   -1.5708
>> sort(v)
```

```
ans =
          0 - 1.0000i     1.0000                    0 + 1.0000i    -1.0000
```

8．prod 函数

功能：元素乘积。其调用格式如下：

B = prod(A)：如果 A 为向量，则计算其元素全部乘积；如果 A 为矩阵，则每列作为向量处理，返回一个每列元素积的行向量；如果 A 为多维数组，则沿第一个非单元素维进行处理，返回元素乘积数组。

B = prod(A,dim)：沿 dim 指定维，返回元素积。

9．cumprod 函数

功能：计算元素的累乘积。其调用格式如下：

B = cumprod(A)：沿数组不同维，返回累乘积，返回值 B 与 A 大小一样，与元素全乘积不同，它只将 X 中相应元素与其之前的所有元素相乘。当 X 是向量，返回 X 的元素累计积向量；如果 X 为矩阵，返回一个与 A 大小相同的每列累乘积的矩阵；如果 A 为多维数组则沿第一个非单元素维计算累乘积。

B = cumprod(A,dim)：返回沿 dim 指定的维元素的累乘积。

cumsum 函数为计算元素的累积和，其用法与 cumprod 函数类似。

【例 1-28】计算元素的乘积、累乘积、累乘和。

```
>> A=[1 2 3;4 5 6;7 8 9];
>> B=prod(A)                %乘积
B =
     28     80    162
>> C=cumprod(A)          %累乘积
C =
      1      2      3
      4     10     18
     28     80    162
>> C=cumprod(A,2)
C =
      1      2      6
      4     20    120
      7     56    504
>> D=cumsum(A)          %累乘和
D =
      1      2      3
      5      7      9
     12     15     18
>> D=cumsum(A,2)
D =
      1      3      6
      4      9     15
      7     15     24
```

1.8　MATLAB 的帮助系统

有效地使用帮助系统中所提供的信息，是用户掌握好 MATLAB 应用的最佳途径。熟练的程序开发人员总会充分地利用软件所提供的帮助信息，而 MATLAB 的一个突出优点就是其拥有较为完善的帮助系统。MATLAB 的帮助系统可以分为联机帮助系统和命令窗口查询帮助系统。

1.8.1　联机帮助系统介绍

1．进入联机帮助系统

MATLAB R2009a 的联机帮助系统非常全面，几乎包括该软件的所有内容。可以选择如下 4 种方式进入 MATLAB R2009a 的帮助系统：

（1）在 MATLAB 主窗口中，单击【Help】菜单下的【Product Help】命令。

（2）按“F1”快捷键，系统将弹出帮助窗口。

（3）按主窗口工具栏中的“?（帮助）”按钮进入帮助窗口。

（4）在命令窗口中直接输入“helpwin”命令、“helpdesk”命令或“doc”命令进入帮助窗口。

2．帮助导向界面

在联机帮助系统中，左侧部分为帮助导向界面，右侧为帮助显示界面。

帮助导向界面下侧的 4 个选项卡分别为 Contents（帮助主题）、Index（帮助索引）、SearchResults（帮助查询）和 Demos（联机演示）。下面分别介绍它们的用法。

1）Contents（帮助主题）

单击该选项卡，将提示 MATLAB R2009a 的帮助内容，如图 1-12 所示。在 MATLAB R2009a 弹出的帮助窗口中，左侧相当于一个目录系统，列出了 MATLAB R2009a 帮助系统中所包含的各项主要内容，如图 1-12 所示。单击其中的任意一项，窗口的右侧将显示该项内容的具体解释。初始状态时，右侧显示的内容是关于帮助系统的一些介绍，指导用户对帮助系统进行充分的运用。

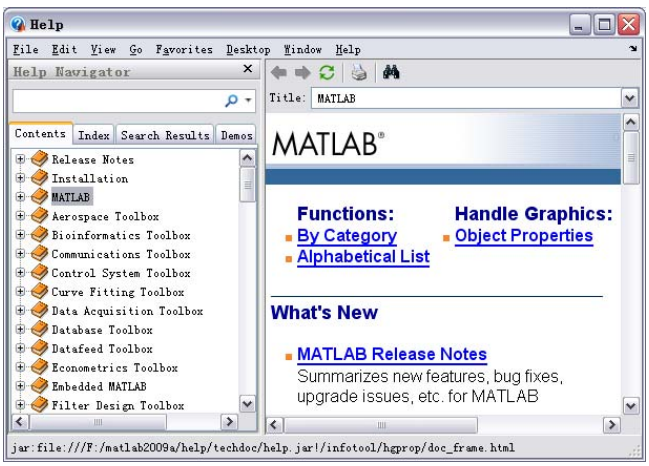


图 1-12　联机帮助系统

2）Index（帮助索引）

单击该选项卡，在“Enter index term”文本框中输入用户需要查找的内容，右边的窗口中

会显示关于该内容的相关信息。如用户要学习 loglog 函数的用法，可以在“Enter index term”文本框中输入“loglog”命令，按回车键，右侧窗口中立即显示出相关的信息，如语法、描述与其相关的函数等。

3）SearchResults（帮助查询）

单击该 Search 选项卡，在“Search for”下拉列表框中选择或输入文件名。如在本例中输入函数名“loglog”，单击“Go”按钮，就会在右侧窗口中输出相关文件和 loglog 的相关信息。注意，虽然使用帮助索引和帮助查询都可以得知某个具体函数的使用方法，但是两者还是有区别的。在使用帮助索引时，左侧显示的信息是按字母排序的，这些信息很多与 loglog 函数并无关系。而使用帮助查询时，右侧显示的是与 loglog 函数相关的一些信息。使用这些信息比上例中使用 Index 所查询的信息要丰富得多。

4）Demos（联机演示）

MATLAB 除了常规的帮助系统外，还设立了联机演示系统，对于初学者来说，查看 MATLAB 的联机演示是最佳的学习方法。在该项内容中，MATLAB 设置了许多关于各个工具箱内容现成程序，用户可以选择自己所需的部分来学习相关内容。

用户可以使用如下两种方法进入联机演示：

（1）单击帮助窗口中的“Demos”按钮。此时联机演示窗口包含两个部分，左边的部分是项目栏，用户可以来选择所需演示的项目；右边的是对此项目的说明文字。双击左边项目栏的具体内容，或单击该项内容，再单击右边说明框中的“Run this demo”或“Run in the Command Window”或“Open this model”超链接，MATLAB 都将会弹出新的窗口进行联机演示操作。

（2）在命令窗口中运行“Demos”命令，也可以进入 MATLAB 的联机演示界面。

1.8.2 命令帮助系统介绍

当用户对 MATLAB 有一定了解后，可以通过命令窗口中直接输入命令来获得相关的帮助信息，这种获取方式比联机帮助更为快捷。在命令窗口中获取帮助信息的主要命令为 help 函数和 lookfor 函数。

1．help 函数

help 函数有 4 种用法，分别是 help、help+函数名（函数类名）、helpdesk 和 helpwin。下面对前两种作介绍。

1）help 命令

在命令窗口中直接输入“help”命令，会显示当前帮助系统中所包含的所有项目。需要注意的是，用户在输入该命令后，命令窗口只显示当前搜索路径中的所有目录名称。

```
>> help
HELP topics:
My Documents\MATLAB              - (No table of contents file)
matlab\general                   - General purpose commands.
matlab\ops                       - Operators and special characters.
matlab\lang                      - Programming language constructs.
matlab\elmat                     - Elementary matrices and matrix manipulation.
matlab\randfun                   - Random matrices and random streams.
……
```

2）help+函数名（函数类名）

当用户知道某个函数名称，如果要了解该函数的具体用法，只需在命令窗口中输入“help+函数名”。如用户想了解 conv 函数（求相关系数）的具体用法，只需在命令窗口中输入“help conv”，即可得如下关于此函数的基本信息。

```
>> help conv
 CONV Convolution and polynomial multiplication.
    C = CONV(A, B) convolves vectors A and B.  The resulting vector is
    length MAX([LENGTH(A)+LENGTH(B)-1,LENGTH(A),LENGTH(B)]). If A and B are
    vectors of polynomial coefficients, convolving them is equivalent to
    multiplying the two polynomials.
     C = CONV(A, B, SHAPE) returns a subsection of the convolution with size
    specified by SHAPE:
      'full'   - (default) returns the full convolution,
      'same'   - returns the central part of the convolution
                 that is the same size as A.
      'valid' - returns only those parts of the convolution
                 that are computed without the zero-padded edges.
                 LENGTH(C)is MAX(LENGTH(A)-MAX(0,LENGTH(B)-1),0).
     Class support for inputs A,B:
       float: double, single
     See also deconv, conv2, convn, filter and,
    in the signal Processing Toolbox, xcorr, convmtx.
    Overloaded methods:
       gf/conv
    Reference page in Help browser
       doc conv
```

同样，当用户要知道某一函数类型的具体用法，只需在命令窗口中输入“help+函数类名”即可。如果用户要得到 matfun 函数类型的具体用法，只需在命令窗口中输入“help matfun”，即可得到如下的关于此函数的基本信息。

```
>> help matfun
   Matrix functions - numerical linear algebra.
     Matrix analysis.
       norm             - Matrix or vector norm.
       normest          - Estimate the matrix 2-norm.
       rank             - Matrix rank.
       det              - Determinant.
       trace            - Sum of diagonal elements.
       null             - Null space.
       orth             - Orthogonalization.
       rref             - Reduced row echelon form.
       subspace         - Angle between two subspaces.
     Linear equations.
……
cholupdate    - rank 1 update to Cholesky factorization.
qrupdate      - rank 1 update to QR factorization.
```

2．lookfor 函数

一般来说，当用户知道某个函数的具体名称时，可以使用 help 函数寻找到相关的帮助信息。但是对于初学者来说，往往不知道函数的确切名称，在这种情况下使用 lookfor 函数可以很方便地解决这个问题。在使用 lookfor 函数时，用户只需知道某个函数的部分关键字，在命令窗口中输入“lookfor+关键字”，就可以很方便地实现查找。如用户需要查找含有关键字 prod 的相关内容，即可以按照下述的方法来实现。

```
>> lookfor prod
cgslparser              - parse Simulink diagram to produce expression for CAGE feature
beep                    - Produce beep sound.
kron                    - Kronecker tensor product.
cross                   - Vector cross product.
dot                     - Vector dot product.
cumprod                 - Cumulative product of elements.
prod                    - Product of elements.
texlabel                - Produces the TeX format from a character string.
datatipinfo             - Produce short description of input variable
……
ss2th                   - Produces a parameterized state-space model.
messageProductNameKey   - returns messageProductNameKey
mpc555bootver           - translates bootcode version to/from product version
messageProductNameKey   - returns messageProductNameKey
```

第 2 章　信号分析基础

本章首先介绍连续时间信号和离散时间信号转换时应该遵循的基本规律——采样定理；然后介绍信号处理课程中的基本信号，这些基本信号是构造复杂信号的基本单元，分析复杂信号时通常要将其分解为基本信号，因此了解基本信号的性质是掌握信号处理的基础。

2.1　时间信号及采样定理

2.1.1　时间信号

时间信号分为连续时间信号和离散时间信号。

1．连续时间信号

连续时间信号是指在所讨论的时间间隔内，对于任意时间值（除若干个不连续点之外）都可以给出确定的函数值。

下面以地震观测系统（地震仪）的数学表示形式为例，来认识线性连续时间系统的描述。地震仪最核心的部件为摆，地震仪的摆的运动方程可用下面形式的微分方程来表示：

$$\ddot{y} = 2\varepsilon_1 \dot{y} + n_1^2 y = -\alpha \ddot{x} \tag{2-1}$$

式中：x 为输入信号；y 为摆的振动，即输出信号；ε_1、n_1 和α为常数；$\ddot{y}$ 和 $\ddot{x}$ 表示对输出信号 y 和输入信号 x 的二阶导数；$\dot{y}$ 为对输出信号的一阶导数。可以将该微分方程看成下列一般方程的一种特殊形式：

$$\begin{aligned} &a(1)y^{(na)}(t) + a(2)y^{(na-1)}(t) + \cdots + a(na+1)y(t) \\ &= b(1)x^{(nb)}(t) + b(2)x^{(nb-1)}(t) + \cdots + b(nb+1)x(t) \end{aligned} \tag{2-2}$$

式中：$y(t)$和 $x(t)$为系统的输入和输出；$y^{(i)}$（i=1,⋯,na）是系统输出量的 i 阶导数；$x^{(i)}$（i=1,2,⋯,nb）是系统输入量的 i 阶导数；$a(1),\cdots,a(na+1)$，$b(1),\cdots,b(nb+1)$为常系数。

对于上面所讲的地震仪观测系统，其运动方程规范成上面的形式，公式如下：

$$a(1)y^{(2)}(t) + a(2)y^{(1)}(t) + a(3)y(t) = b(1)x^{(2)}(t) \tag{2-3}$$

2．离散时间信号

离散时间信号可由连续时间信号 $x(t)$通过抽样获得，设抽样时间间隔为 T，故用 $x(xT)$表示此离散时间信号在 nT 点上的值，n 为整数。由于离散时间信号处理常常是非实时的，可以先记录数据后分析或存放在存储器中以供随时取用，因此 $x(nT)$可以看作是按照一定顺序排列的一组数据，可以直接用 $x(n)$表示第 n 个离散时间点的序列值，并用$\{x(n)\}$表示离散时间信号——序列。为方便起见，通常情况下直接用 $x(n)$表示离散序列。

离散时间信号——序列。可以用图形来描述，如图 2-1 所示，纵轴线段的长短代表各序列值的大小，横轴代表离散时间点。注意，横轴虽然为连续直线，但是 $x(n)$只有在 n 为整数时才

有意义，n 不是整数时没有意义，但不能认为 $x(n)$的值为零。

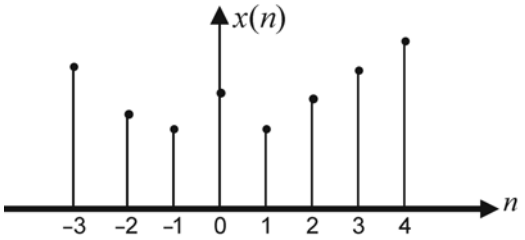


图 2-1　离散时间信号——序列的图形表示

应该注意，离散序列{$x(n)$}可由连续信号 $x(t)$在 nT 时刻采样得到，T 为采样周期。实际应用中，序列为有限的。即一个信号序列表示为{$x(n)$}，n 应满足条件 $N_1 \leqslant n \leqslant N_2$，$N_1$、$N_2$ 均为正整数。

MATLAB 中采用向量表示序列。由于 MATLAB 矢量的第一个元素位置为 $x(1)$，因此，为了清楚地表示序列{$x(n)$}要用到两个向量：一个向量 n 表示序列元素的位置（称为序号序列），而另一个向量 x 表示序列值（称为值序列）。为了在图形中表示这些向量，在 MATLAB 中经常用 stem 函数和 plot 函数。

【例 2-1】stem 函数示例。

其实现的 MATLAB 程序代码如下：

```
>> x = 0:25;
y = [exp(-.07*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
set(h(1),'MarkerFaceColor','blue');
set(h(2),'MarkerFaceColor','red','Marker','square');
```

运行程序，效果如图 2-2 所示。

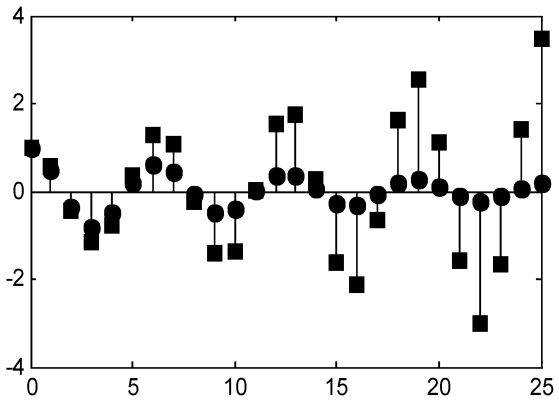


图 2-2　stem 函数产生的序列

【例 2-2】设一个信号值为{0 1 2 3 2 1 0 −1 −2 −3 −4 −5 1 2}，序号自-3 到 10，试用 MATLAB 中的 stem 函数和 plot 函数表示这个信号，其中信号的采样间隔假定为 1s。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=[-3 -2 -1 0 1 2 3 4 5 6 7 8 9 10];   %为序号序列
X=[0 1 2 3 2 1 0 -1 -2 -3 -4 -5 1 2];  %为值序列
```

```
subplot(2,1,1);stem(N,X);              %绘制离散值图
hold on;
plot(N,zeros(1,length(X)),'r');
%绘制横轴,zeros(1,N)为产生 1 行 N 列元素值为零的数组
set(gca,'box','on');                   %产生坐标轴设在方框上
xlabel('序列号');ylabel('序列值');
title('stem 例子');
dt=1;                                  %时间间隔
t=N*dt;                                %时间序列
subplot(2,1,2);plot(t,X);              %绘制随时间的变化
hold on;
plot(t,zeros(1,length(X)),'r');        %绘出横轴
xlabel('时间/s');ylabel('函数值');
title('plot 例子');
```

运行程序，效果如图 2-3 所示。

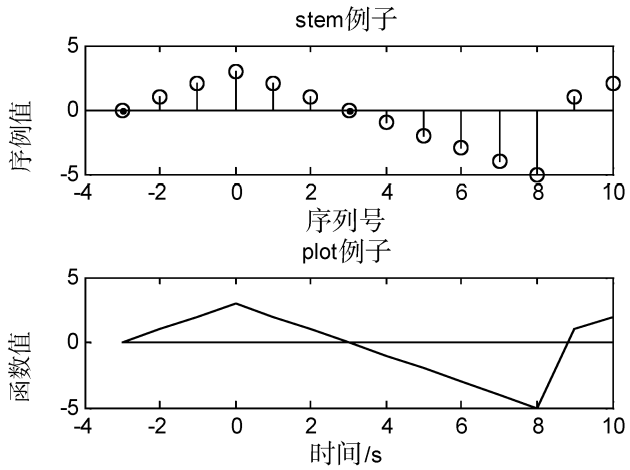


图 2-3　时间信号的表示

由上例可以看到，stem 函数清楚地表示了离散信号。用 plot 函数可以绘出数学中所讲的函数。如果样点足够多，可以清楚地反映函数的变化规律。

对于一个连续时间信号，$x(t)$可以每隔一定的时间间隔量取一个值，将这些值组合成一个序列，就将连续信号离散化为离散时间信号$\{x(n)\}$。这样得到的离散时间信号能否不失真地反映连续时间信号呢？这里涉及一个频率折叠和假频问题。下面将讨论这个问题。

2.1.2　采样定理

后面将讲到，每个信号都可以分解为多个不同频率、不同振幅和不同相位的正弦或余弦的函数叠加的形式。为了讨论方便，假定要离散化的信号只有一个周期成分。

如果此信号以每周期少于两个点采样，会发生什么情况呢？其实这个效应多少类似于频闪光照射一个高速旋转轮子时所观测到的效应。若照射的速率与轮辐通过某一参考点的速率不同，这个轮子好像以不同于真速率的速度在旋转。下面用 MATLAB 模拟来研究这种情况。

【例 2-3】现有一个振幅为 1、频率为 10Hz、相位为 0.3 的模拟信号，即 $\sin(2\pi\times10\times t+0.3)$，用 0.01s 的采样间隔（采样频率为 100Hz）来表示原始信号（实际上模拟信号不能用离散值表示，此处为了在计算机上表示，用采样率非常高的离散信号表示模拟信号）。若每秒采样 10 次，即采样间隔为 0.1s，试绘出原始信号和采样后的信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
dt=0.01;n=0:90-1;
t=n*dt;
f=10;                          %原始信号的频率为 10Hz
x=sin(2*pi*f*t+0.3);           %在计算机上的原始信号
dt=0.1;
n=0:10-1;
t1=n*dt;
% 以 10Hz 的采样频率采样,为取一样的时间长度
% 序号长度为原始信号序号长度的 1/10
x1=sin(2*pi*f*t1+0.3);    %采样后的信号
subplot(3,1,1);plot(t,x);
% 绘出模拟原始信号,为与下图统一,采样 y 轴的范围[-1 1]用 ylim 给出
ylim([-1,1]);
title('原始信号');
% 绘出在模拟信号基础上的采样过程
subplot(3,1,2);plot(t,x,t1,x1,'rp');
ylim([-1,1]);
title('采样过程');
%绘出采样后的信号
subplot(3,1,3);plot(t1,x1);
ylim([-1,1]);xlabel('时间/s');
title('采样后信号');
```

运行程序，效果如图 2-4 所示。

可以说，采过样的 10Hz 的信号具有“零频率”信号的外貌（图 2-4）。拿旋转的轮子来类比，闪光速率为 10Hz 的照相使得轮子看起来好像静止不动。在这个例子中，一个 10Hz 的信号表现为一个“零频率”信号，这就是采样过程造成的。再来考虑一个频率 9Hz 的原始正弦信号，以 0.1s 间隔的采样结果产生的信号具有 1Hz 信号的外貌（图 2-5）。这个过程可以将上述程序中的 f=10 改为 f=9，重新运行即可得到图 2-5 所示的效果。这类出现与真实频率不一致的现象称为“假频”。“假频”有什么规律可循呢？如果按表 2-1 多做几次试验，分别观察 f=8Hz、7Hz、6Hz，会发现：当信号频率低于采样频率的 1/2 时（在此处为 5Hz），采样过的信号可以反映原始信号的特征（可以修改上述程序观看效果）；当信号频率超过采样频率的 1/2 时，就会出现数字采样过的信号与原来的信号不一致。其表现的信号频率为采样频率（此处为 10Hz）减去原始信号的频率（此处为10Hz～9Hz）。这有点像儿时玩的折纸，即将原始信号中的频率以采样频率的 1/2 折叠到低频中所表现的频率。当信号频率超过 10Hz 时又如何呢？仍采用上面的 MATLAB 程序进行试验，改为 f=11，12，13，14，…，20 等值，会发现，这些信号采样后所表现的信号分别与 f=1，2，3，4，…，10 所表现的频率完全一致，即出现了循环，更高的频率与此类似。

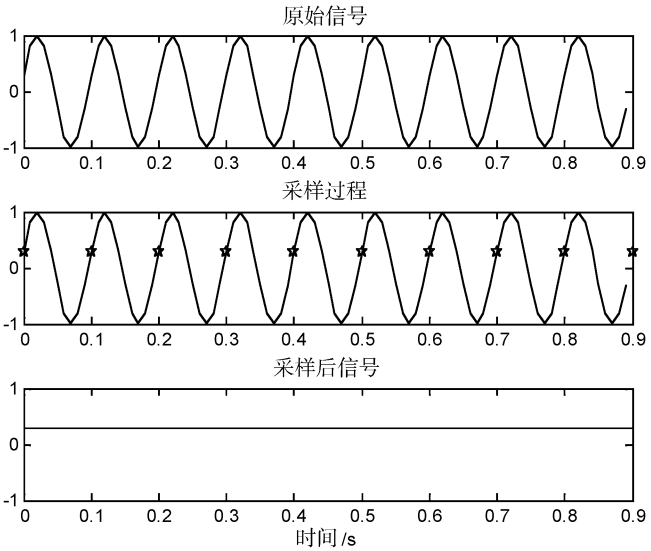


图 2-4　10Hz 的正弦信号经 10Hz 的采样频率后的效果

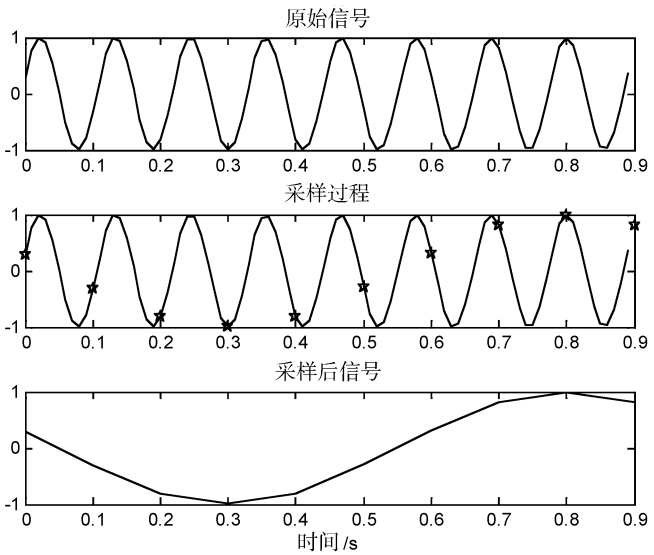


图 2-5　9Hz 的正弦信号经 10Hz 的采样频率后的效果

表 2-1　以采样周期为 0.1s（采样频率为 10Hz）对信号采样试验

原信号频率/Hz	采样频率/Hz	采样后视频率/Hz
f=10	10	0
f=9	10	1
f=8	10	2
f=7	10	3
f=6	10	4
f=5	10	5

续表

原信号频率/Hz	采样频率/Hz	采样后视频率/Hz
f=4	10	4
f=3	10	3
f=2	10	2
f=1	10	1
f=11	10	1
f=12	10	2
f=13	10	3
f=14	10	4
f=15	10	5
f=16	10	4
f=17	10	3
f=18	10	2
f=19	10	1

通过上面的分析可以得到：当采样频率大于信号中所含有信号最大频率的两倍时，采样后的数据可以不失真地描述信号。当采样频率不满足这个条件时，会出现频率折叠和频率重复。这就是采样定理。

在数字信号处理中通常定义采样频率的 1/2 为 Nyquist（奈奎斯特）频率。因此，采样定理还可以叙述如下：只有信号中的最大频率不大于奈奎斯特频率，采样后的数据才能不失真地反映信号。

2.2 信号的产生

在 MATLAB 中对基本信号的产生提供了许多相应的函数，下面将对一些基本信号的产生作简单介绍。

1．chirp 函数

功能：线性调频信号发生器。其调用格式如下：

y = chirp(t,f0,t1,f1)：产生一个线性扫频（频率随时间线性变化）采样信号，其时间轴的设置由数组 t 定义。时刻 0 的瞬时频率为 f0；时刻 t1 的瞬时频率为 f1。默认情况下，f0=0Hz，t1=1，f1=100Hz。

y = chirp(t,f0,t1,f1,'method')：指定改变扫频的方法。可用的方法有“linear”（线性调频），“quadratic”（二次调频）和“logarithmic”（对数调频）；默认为“linear”。注意：对于对数扫频，必须有 f1>f0。

y = chirp(t,f0,t1,f1,'method',phi)：指定信号的初始相位为 phi（单位为度），默认时 phi=0。

y = chirp(t,f0,t1,f1,'quadratic',phi,'shape')：指定形状的二次扫频信号的频谱图。形状或凸或凹，如果为 downsweep（f0>f1），即凸形，如果为 upsweep（f0<F1），即凹形。默认为凸形。凹凸形状如图 2-6 所示。

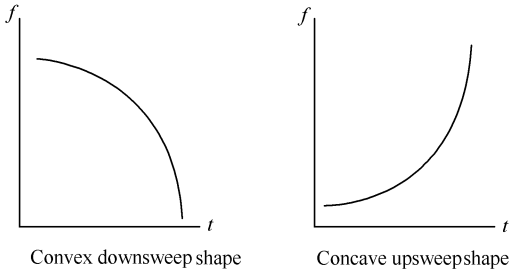


图 2-6　凹凸形状

【例 2-4】在不同的采样时间下计算谱图与线性调频信号瞬时频率偏差。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t = 0:0.001:2;
y = chirp(t,0,1,150);
subplot(2,3,1);spectrogram(y,256,250,256,1E3,'yaxis');
xlabel('t=0:0.001:2 采样时间');
t = -2:0.001:2;
y = chirp(t,100,1,200,'quadratic');
subplot(2,3,2);spectrogram(y,128,120,128,1E3,'yaxis');
xlabel('t=-2:0.001:2 采样时间');
t = -1:0.001:1;
fo = 100; f1 = 400;
y = chirp(t,fo,1,f1,'q',[],'convex');
subplot(2,3,3);spectrogram(y,256,200,256,1000,'yaxis')
xlabel('t=-1:0.001:1 采样时间');
t = 0:0.001:1;
fo = 100; f1 = 25;
y = chirp(t,fo,1,f1,'q',[],'concave');
subplot(2,3,4);spectrogram(y,hanning(256),128,256,1000,'yaxis');
xlabel('t=0:0.001:1 采样时间');
t = 0:0.001:10;
fo = 10; f1 = 400;
y = chirp(t,fo,10,f1,'logarithmic');
subplot(2,3,6);spectrogram(y,256,200,256,1000,'yaxis')
xlabel('t=0:0.001:10 采样时间');
```

运行程序，效果如图 2-7 所示。

2．diric 函数

功能：周期 sinc 函数（Dircichlet 函数）发生器。其调用格式如下：

y = diric(x,n)：返回一大小与 x 相同的矩阵，其元素为 Dirichlet 函数。n 必须为正整数，该函数将 0～2π等间隔分成 N 等份。

dirichlet 函数的定义如下：

$$d(x)=\frac{\sin(nx/2)}{n\sin(x/2)}$$

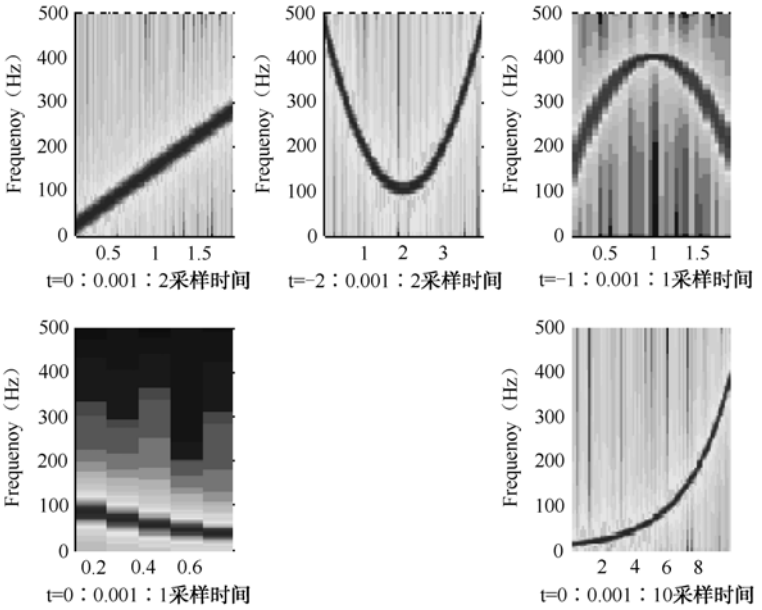


图 2-7　线性调频信号

【例 2-5】diric 函数示例。

```
>> clear all;
x=-1:0.001:1;
y=diric(x,1000);
plot(y);
xlabel('采样时间');ylabel('幅值');
```

运行程序，效果如图 2-8 所示。

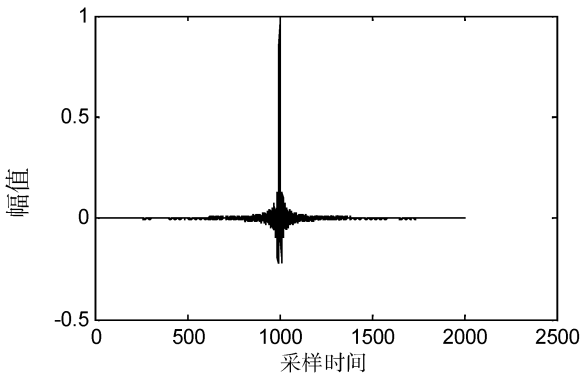


图 2-8　diric 信号图

3．gauspuls 函数

功能：高斯函数调幅的正弦波形发生器。其调用格式如下：

yi = gauspuls(t,fc,bw)：返回最大幅值为 1 的高斯函数调幅的正弦波的采样，其中心频率为 fc（单位为 Hz），相对带宽为 bw，时间由数组 t 给定。注意：bw 必须大于 0。默认时，fc=1000Hz，bw=0.5。

yi = gauspuls(t,fc,bw,bwr)：指定可选的频带边缘处的参考水平 bwr，以相对于正常信号峰值

下降了−bwr（单位为 dB）为边界的频带，其相对带宽为 100*bw%，默认时，bwr= −6dB。注意 bwr 必须为负数。

[yi,yq] = gauspuls(...)：同时返回同相及其积分信号。

[yi,yq,ye] = gauspuls(...)：同时返回信号的包络 ye。

tc = gauspuls('cutoff',fc,bw,bwr,tpe)：返回包络相对包络峰值下降 tpe（单位为 dB）时的时间 tc（>=0），默认时 tpe= −60dB。注意：tpe 必须小于 0。

【例 2-6】绘一中心频率为 50kHz 的高斯型正弦脉冲，其带宽为 60%，采样率为 1MHz；信号在包络相对于峰值下降 40dB 时截断的信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
tc = gauspuls('cutoff',50e3,0.6,[],-40);
t = -tc : 1e-6 : tc;
yi = gauspuls(t,50e3,0.6);
plot(t,yi);
xlabel('采样时间');ylabel('幅值');
```

运行程序，效果如图 2-9 所示。

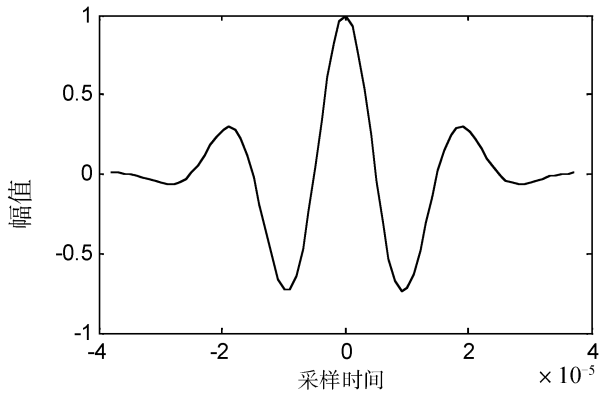


图 2-9　gauspuls 信号图

4．pulstran 函数

功能：脉冲序列的发生器。其调用格式如下：

y = pulstran(t,d,'func')：返回的是对一连续函数‘func’进行采样而得到的脉冲序列。函数‘func’的输入参数 t 给定的值减去偏移量 d，这样函数求 length(d)次值，其值由 y 返回。y=func(t−d(1))+func(t−d(2))+…。注意：函数‘func’必须能接受数组 t 作为其输入参数。当 d 是一个列数为 2 的向量时，可给上面的求和式中的每一项加一增益系数，其中 d 的第一列为延迟量，第二列为相应的增益系数。注意：一行向量将被认为是延迟参数。可通过 pulstran(t, d, 'func', p1, p2, ...)给函数‘func’传送必要的参数，函数的调用形式为 func(t –d(1), p1, p2,...)。

pulstran(t,d,p,fs)：产生一脉冲序列，序列以 fs 的采样率，由对 p 给定的脉冲原型的多项延迟插值之和进行采样得到的。p 被设定为时间间隔[0，(length(p)−1)/fs]之内，在此间隔之外，采样数据都为 0。默认情况下，延迟由线性插值产生。

pulstran(t,d,p)：认为 fs=1Hz。

pulstran(...,'func')：指定另外的插值方法。

【例 2-7】绘制一个在 10kHz，通带 50%的周期高斯脉冲波形，其重复频率为 1kHz，锯齿宽度为 0.01s，衰减率为 0.8，采样率为 50kHz。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t = 0 : 1/50E3 : 10e-3;
d = [0 : 1/1E3 : 10e-3 ; 0.8.^(0:10)]';
y = pulstran(t,d,'gauspuls',10e3,0.5);
plot(t,y)
xlabel('采样时间');ylabel('幅值');
```

运行程序，效果如图 2-10 所示。

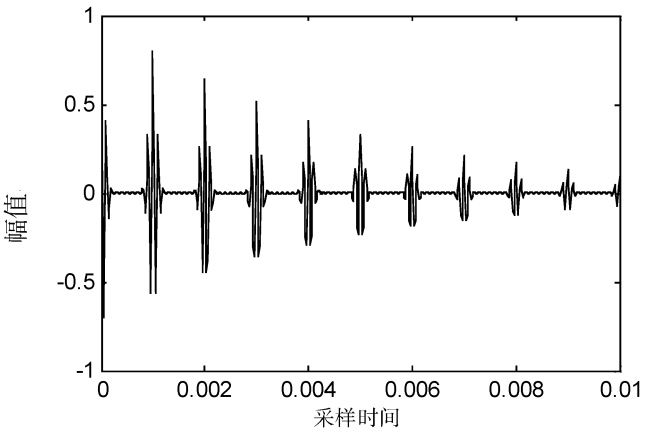


图 2-10　pulstran 信号图

5．square 函数

功能：方波发生器。其调用格式如下：

x = square(t)：返回一周期为 2π的方波，采样的时刻由向量 t 指定。square(t)与 sin(t)差不多，只不过其产生的是峰值为−1～1 的方波而非正弦波。

x = square(t,duty)：产生一给定占空比的方波。占空比 duty 是信号为正值的比例。

【例 2-8】利用 square 函数产生一个方波。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0:0.0001:0.625;
y=square(2*pi*30*t,0.5);
plot(t,y);
set(gco,'linewidth',1);
set(gca,'fontname','宋体','fontsize',10);
xlabel('采样时间','fontname','宋体','fontsize',10);
ylabel('幅值','fontname','宋体','fontsize',10);
```

运行程序，效果如图 2-11 所示。

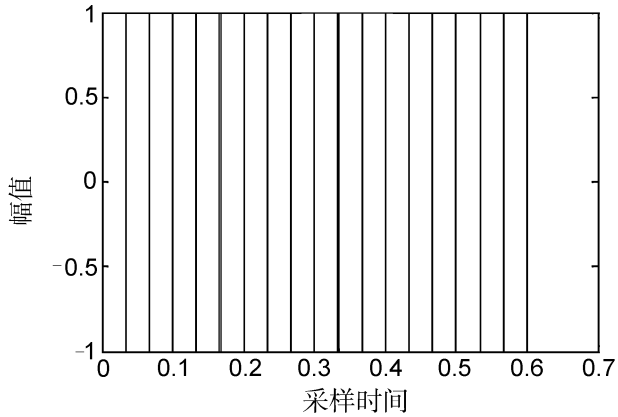


图 2-11　方波信号图

6．sawtooth 函数

功能：锯齿波和三角波发生器。其调用格式如下：

sawtooth(t)：产生一个周期为 2π的锯齿波，采样的时刻由向量 t 指定。sawtooth(t)与 sin(t)相似，只不过其产生的是峰值为−1～1 的锯齿波而非正弦波。

sawtooth(t,width)：产生三角波，width 指定最大值出现的位置，其取值为 0～1，1 对应于 2π。当 t 由 0 增大到 width*2π时，函数值由−1 增大到 1，当 t 由 width*2π增大到 2π时，函数值由 1 减小到−1。因此当 width=0.5 时，产生的是一关于时刻π对称的、峰值为 1 的三角波。

【例 2-9】产生三角波。

```
>> clear all;
t=-1:0.001:1;
x=sawtooth(2*pi*10*t);
plot(t,x);
xlabel('采样时间');ylabel('幅值');
```

运行程序，效果如图 2-12 所示。

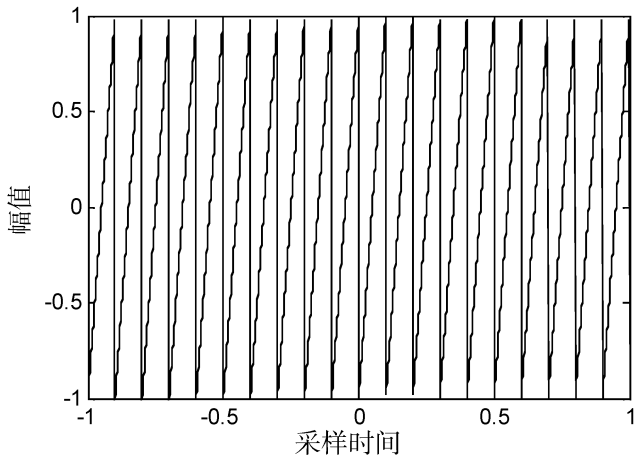


图 2-12　三角波形信号

7．rectpuls 函数

功能：非周期矩形波发生器。其调用格式如下：

y = rectpuls(t)：返回由数组 t 给定时刻的连续、非对称、单位高度的矩形波的采样。矩形波的中心为 t=0；默认时，矩形宽度为 1；注意边界值，如：rectpuls(−0.5)=1 而 rectpuls(0.5)=0。

y = rectpuls(t,w)：产生一宽度为 w 的矩形波。

【例 2-10】产生非周期矩形波。

```
>> clear all;
t=-1:0.001:1;
y=rectpuls(t,0.5);
plot(y);
set(gco,'linewidth',1);
set(gca,'fontname','宋体','fontsize',10);
xlabel('采样时间','fontname','宋体','fontsize',10);
ylabel('幅值','fontname','宋体','fontsize',10);
```

运行程序，效果如图 2-13 所示。

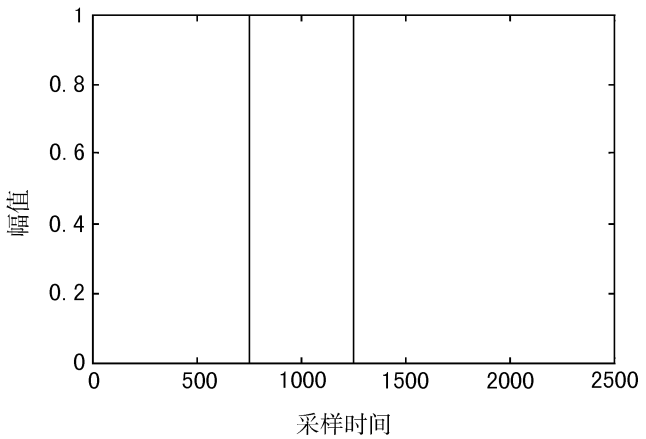


图 2-13　rectpuls 信号图

8．sinc 函数

功能：sinc 函数发生器。其调用格式如下：

y = sinc(x)：返回一个由 sinc 函数值为元素的矩阵。

sinc(x)函数的原型为

$$y=\frac{\sin(\pi x)}{\pi x}$$

【例 2-11】sinc 函数发生器示例。

```
>> clear all;
t = (1:10)';
randn('state',0);
x = randn(size(t));
ts = linspace(-5,15,600)';
y = sinc(ts(:,ones(size(t))) - t(:,ones(size(ts)))')*x;
plot(t,x,'o',ts,y)
set(gco,'linewidth',1);
set(gca,'fontname','宋体','fontsize',10);
xlabel('采样时间','fontname','宋体','fontsize',10);
```

```
ylabel('幅值','fontname','宋体','fontsize',10);
```

运行程序，效果如图 2-14 所示。

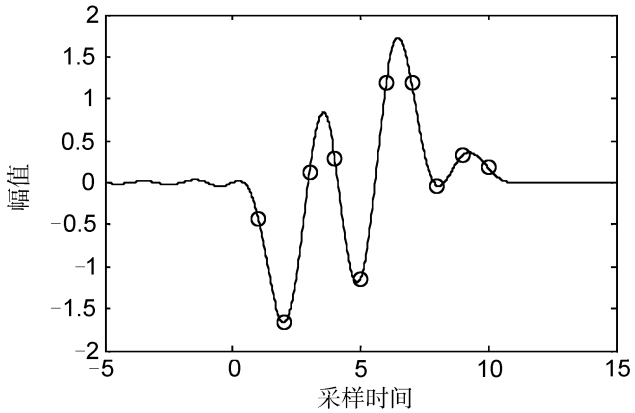


图 2-14　sinc 信号图

9．tripuls 函数

功能：非周期三角脉冲发生器。其调用格式如下：

y = tripuls(T)：产生一连续、非周期、单位高度的三角脉冲的采样，采样时刻由数组 T 指定。默认时，三角脉冲是非对称的，且其宽度为 1。

y = tripuls(T,w)：产生一宽度为 w 的三角脉冲。

y = tripuls(T,w,s)：允许调整三角形的斜度 s。参数 s 必须满足−1<s<+1，当 s=0 时，产生一个对称三角脉冲。

【例 2-12】tripuls 函数示例。

```
>> clear all;
fs = 10000;
t = -1:1/fs:1;
w = .4;
x = tripuls(t,w);
figure,plot(t,x)
xlabel('采样时间','fontname','宋体','fontsize',10);
ylabel('幅值','fontname','宋体','fontsize',10);
```

运行程序，效果如图 2-15 所示。

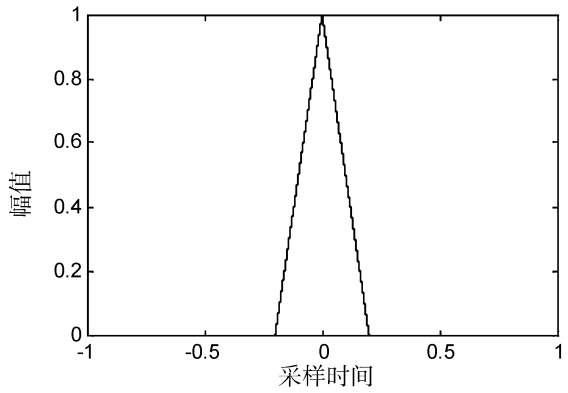


图 2-15　三角脉冲信号图

2.3 连续时间信号在 MATLAB 中的运算

2.3.1 信号的时移、反折和尺度变换

信号的时移、反折和尺度变换是针对自变量时间而言的，其数学表达式与波形变化之间存在一定的变化规律。

信号 $f(t)$的时移就是将信号数学表达式中的自变量 t 用 $t\pm t_0$ 替换，其中 t_0 为正实数。因此，波形的时移变换是将原来的 $f(t)$波形在时间轴上向左或向右移动。$f(t+t_0)$为 $f(t)$波形向左移动 t_0；$f(t-t_0)$为 $f(t)$波形向右移动 t_0。信号 $f(t)$的反折就是将表达式中的自变量 t 用 $-t$ 替换。波形变换后，$f(-t)$的波形是原来的 $f(t)$相对于纵轴的镜像。信号 $f(t)$的尺度变换就是将表达式中的自变量 t 用 at 替换，其中 a 为正实数。对应于波形的变换，则是将原来的 $f(t)$波形以原点为基准压缩（$a>1$）至原来的 $1/a$，或者扩展（$0<a<1$）至原来的 $1/a$。

综合上述三种情况，如果将信号 $f(t)$的自变量 t 用 $t\pm t_0$ 替换，其中，a、t_0 为实数，则 $f(at+t_0)$相对于 $f(t)$或者扩展（$|a|<1$）或者压缩（$|a|>1$）；或者反折（$a<0$）或者时移（$t_0\neq 0$），而波形仍保持与原 $f(t)$相似的形状。利用 MATLAB 可方便直观地观察和分析信号的时移、反折和尺度变换对信号波形的影响。

【例 2-13】已知信号 $f(t)$的波形如图 2-16 所示，试用 MATLAB 命令画出 $f(t-2)$、$f(3t)$、$f(-t)$、$f(-3t-2)$的波形图。

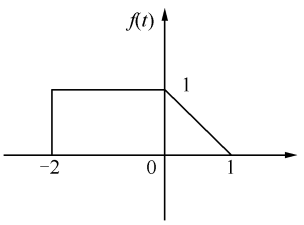


图 2-16　$f(t)$的波形

根据图 2-16 中 $f(t)$的波形，先建立 $f(t)$函数。

```
function f=funct1(t)
f=uCT(t+2)-uCT(t)+(-t+1).*(uCT(t)-uCT(t-1));

function f=uCT(t)
f=(t>=0);
```

其实现的 MATLAB 源代码如下：

```
>> clear all;
t=-2:0.01:4;
ft1=funct1(t-2);
subplot(2,2,1);plot(t,ft1);
xlabel('(a) f(t-2) ');
grid on;
axis([-2 4 -0.5 2]);
ft2=funct1(3*t);
subplot(2,2,2);plot(t,ft2);
```

```
xlabel('(b) f(3t) ');
grid on;
axis([-2 4 -0.5 2]);
ft3=funct1(-t);
subplot(2,2,3);plot(t,ft3);
xlabel('(c) f(-t) ');
grid on;
axis([-2 4 -0.5 2])
ft4=funct1(-3*t-2);
subplot(2,2,4);plot(t,ft4);
xlabel('(d) f(-3t-2) ');
grid on;
axis([-2 4 -0.5 2])
```

运行程序，效果如图 2-17 所示。

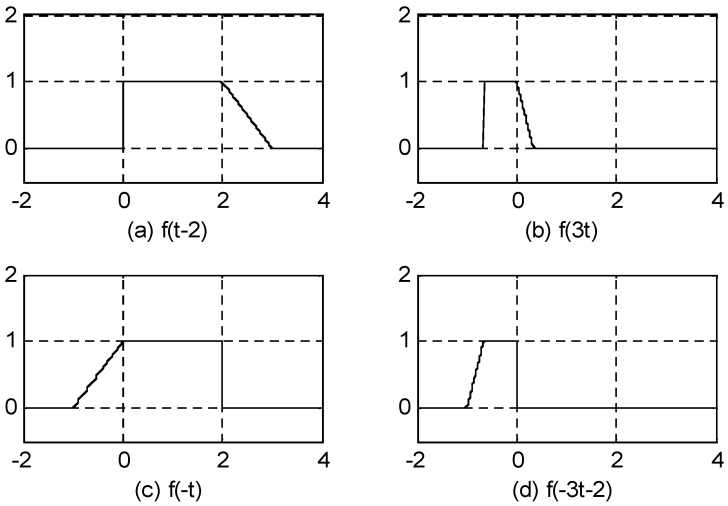


图 2-17　相应信号的波形图

2.3.2　连续时间信号的微分与积分运算

1．连续时间信号的微分运算

对于连续时间信号，其微分运算如果用符号表达式表示，则用 diff 命令函数可完成求导运算，其调用格式如下：

diff(function, 'variable', n)

其中：function 表示需要进行求导运算的函数，或者被赋值的符号表达式；variable 为求导运算的独立变量；n 为求导阶数，默认值为求一阶导数。

【例 2-14】用 MATLAB 命令求下列函数关于变量 x 的一阶导数。

（1）$y_1=\sin(ax^2)$；（2）$y_2=\sin x1nx$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
syms a x y1 y2
y1=sin(a*x^2);
```

```
y2=x*sin(x)*log(x);
dy1=diff(y1,'x')
dy1 =
        2*a*x*cos(a*x^2)
>> dy2=diff(y2)
dy2 =
        sin(x) + log(x)*sin(x) + x*cos(x)*log(x)
```

2．连续时间信号的积分运算

连续时间信号的积分运算如果用符号表达式来表示，用 int 命令函数可完成积分运算，其调用格式如下：

```
int(function, 'variable', a, b)
```

其中：function 表示被积函数，或者被赋值的符号表达式；variable 为积分变量；a 为积分下限，b 为积分上限；a 和 b 默认时则求不定积分。

【例 2-15】用 MATLAB 命令计算不定积分 $\int\left(x^5-ax^2+\frac{\sqrt{2}}{2}\right)\mathrm{d}x$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
syms a x y3
y3=x^5-a*x^2+sqrt(x)/2;
int(y3,'x')
```

运行程序，输出如下：

```
ans =
x^(3/2)/3 - (a*x^3)/3 + x^6/6
```

【例 2-16】用 MATLAB 命令计算定积分 $\int_0^1\frac{x\mathrm{e}^x}{(1+x)^2}\mathrm{d}x$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
syms x y4
y4=(x*exp(x))/(1+x)^2;
int(y4,0,1)
```

运行程序，输出如下：

```
ans =
exp(1)/2 - 1
```

2.3.3 信号的相加与相乘运算

信号的相加与相乘是指在同一时刻信号取值的相加与相乘。因此，MATLAB 对于时间信号的相加与相乘都是基于向量的点运算。

【例 2-17】已知 $f_1(t)=\sin\Omega t$，$f_2(t)=\sin 8\Omega t$，试用 MATLAB 命令绘出 $f_1(t)+f_2(t)$和 $f_1(t)f_2(t)$的波形图，其中，$f=\frac{\Omega}{2\pi}=1\mathrm{Hz}$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
f=1;
t=0:0.01:3/f;
f1=sin(2*pi*f*t);
f2=sin(2*pi*8*f*t);
subplot(2,1,1);
plot(t,f1+1,':',t,f1-1,':',t,f1+f2);
grid on;
xlabel('(a) f1(t)+f2(t)');
subplot(2,1,2);
plot(t,f1,':',t,-f1,':',t,f1.*f2);
grid on;
xlabel('(b) f1(t)*f2(t)');
```

运行程序，效果如图 2-18 所示。

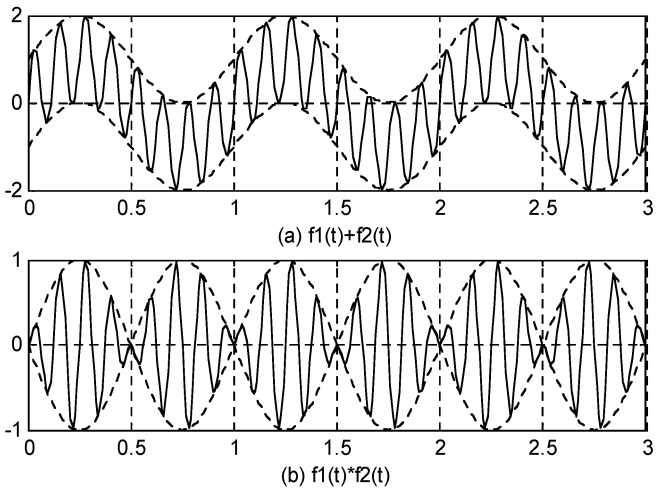


图 2-18　信号的相加相乘波形图

2.3.4　信号的奇偶分解

任何一个函数都可以分解为一个偶函数分量与一个奇函数分量之和的形式，即

$$f(t) = f_e(t) + f_o(t)$$

式中

$$f_e(t) = \frac{1}{2}[f(t) + f(-t)]$$

$$f_o(t) = \frac{1}{2}[f(t) - f(-t)]$$

从波形角度看，求信号的偶分量和奇分量时，首先是将信号进行反折，得到 $f(-t)$，然后与原信号 $f(t)$进行相加减，再除以 2，即可分别得到偶分量 $f_e(t)$和奇分量 $f_o(t)$。

【例 2-18】已知 $f(t)$的波形如图 2-19 所示，用 MATLAB 命令画出 $f(t)$的奇分量和偶分量。

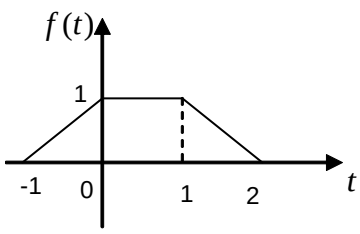


图 2-19　$f(x)$信号的波形

从图 2-19 中 $f(t)$的波形知：

$$f(t)=(t+1)[u(t+1)-u(t)]+[u(t)-u(t-1)]+(2-t)[u(t-1)-u(t-2)]$$
$$=(t+1)u(t+1)-tu(t)-(t-1)u(t-1)+(t-2)u(t-2)$$

根据奇分量和偶分量的公式，可先求出反折信号 $f(-t)$。为图形直观起见，取时间为左右对称，本例取 t=−3:0.01:3。这样，反折信号 $f(-t)$的获得，只要将 $f(-t)$取样值左右对换就可以了，即用 MATLAB 语句 x1=fliplr(x)实现信号反折。信号奇偶分解的 MATLAB 程序代码如下：

```
>> clear all;
t=-3:0.01:3;
f=(t+1).*uCT(t+1)-t.*uCT(t)-(t-1).*uCT(t-1)+(t-2).*uCT(t-2);
subplot(3,1,1);plot(t,f);
grid on;
axis([-3 3 0 1.2]);
xlabel('(a) f(t)');
f1=fliplr(f);
fe=(f+f1)/2;
fo=(f-f1)/2;
subplot(3,1,2);plot(t,fe);
grid on;
axis([-3 3 0 1.2]);
xlabel('(b) fe(t)');
subplot(3,1,3);plot(t,fo);
grid on;
axis([-3 3 -0.6 0.6]);
xlabel('(c) fo(t)');
```

运行程序，效果如图 2-20 所示。

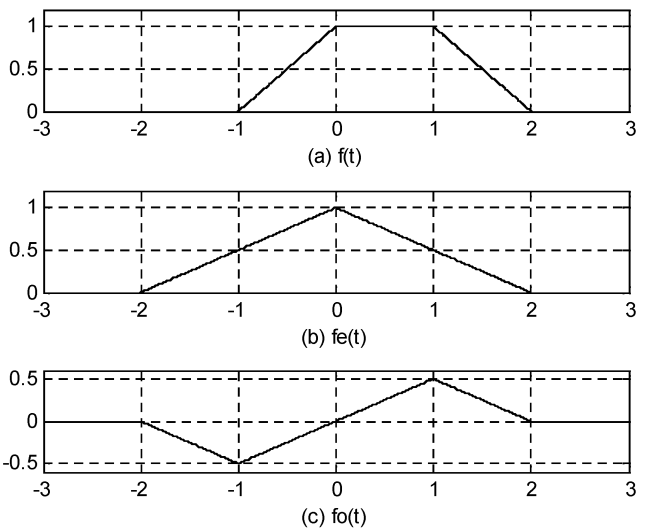


图 2-20　信号的奇偶分解波形

2.4　连续时间 LTI 系统的时域分析

2.4.1　连续时间系统零输入响应和零状态响应的符号求解分析

LTI 连续系统可用线性常系数微分方程来描述，即

$$\sum_{i=0}^{N} a_i y^{(i)}(t) = \sum_{j=0}^{M} b_j f^{(j)}(t)$$

式中：$a_i(i=0,1,\cdots,N)$和 $b_j(j=0,1,\cdots,M)$为实常数。

该系统的完全响应由零输入响应和零状态响应两部分组成。零输入响应是指输入信号为 0，仅由系统的起始状态作用所引起的响应，通常用 $y_{zi}(t)$表示；零状态响应是指系统在起始状态为 0 的条件下，仅由激励信号作用所引起的响应，通常用 $y_{zi}(t)$表示。

MATLAB 符号工具箱提供了 dsolve 函数，可实现常系统微分方程的符号求解，其调用格式如下：

dsolve('eq1','eq2',...,'cond1','cond2',...,'v')

其中：参数 eq1、eq2、…表示各微分方程，它与 MATLAB 符号表达式的输入基本相同，微分或导数的输入是用 Dy、D2y、D3y、…表示 y 的一阶导数 y′、二阶导数 y″、三阶导数 y‴、…；参数 cond1、cond2、…表示各初始条件或起始条件；参数 v 表示自变量，默认为变量 t。可利用 dsolve 函数来求解系统微分方程的零输入响应和零状态响应，进而求出完全响应。

【例 2-19】试用 MATLAB 命令求齐次微分方程 $y'''(t)+2y''(t)+y'(t)=0$ 的零输入响应，已知起始条件为 y(0_)=1，$y'(0_)=1$，$y''(0_)=2$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
eq='D3y+2*D2y+Dy=0';              %定义符号微分方程表达式
cond='y(0)=1,Dy(0)=1,D2y(0)=2';   %初始条件
ans=dsolve(eq,cond);
simplify(ans)
ans =
     5 - (3*t)/exp(t) - 4/exp(t)
```

在求解该微分方程的零输入响应过程中，0_-到 0_+是没有跳变的，因此，程序中初始条件选择 t=0 时刻，即 cond='y(0)=1,Dy(0)=1;D2y(0)=2'。

【例 2-20】试用 MATLAB 命令求解微分方程 $y''(t)+3y'(t)+2y(t)=x'(t)+3x(t)$，当输入 $x(t)=e^{-3t}u(t)$，起始条件为 $y(0_-)=1$、$y'(0_-)=2$ 时系统的零输入响应，零状态响应及完全响应。

求得零输入和零状态响应后，完全响应则为二者之和。其实现的 MATLAB 程序代码如下：

```
>> clear all;
eq='D2y+3*Dy+2*y=0';              %齐次解求零输入响应
cond='y(0)=1,Dy(0)=2';
yzi=dsolve(eq,cond);
yzi=simplify(yzi)
yzi =
-3*exp(-2*t)+4*exp(-t)
```

```
eq1='D2y+3*Dy+2*y=Dx+3*x';      %零状态响应求解
eq2='x=exp(-3*t)*Heaviside(t)';
cond='y(-0.001)=0,Dy(-0.001)=0';   %起始条件
yzs=dsolve(eq1,eq2,cond);
yzs=simplify(yzs.y)
yzs =
-heaviside(t)*(exp(-2*t)-exp(-t))
>> yt=simplify(yzi+yzs)
yt =
-3*exp(-2*t)+4*exp(-t)-exp(-2*t)*heaviside(t)+exp(-t)*heaviside(t)
```

利用符号求解出零输入响应、零状态响应及完全响应后，可利用 ezplot 命令绘出它们的波形，以便观察。例如，可以分别绘出图 2-21 所示的零输入响应、零状态响应及完全响应，其实现的 MATLAB 程序代码如下：

```
>> subplot(3,1,1);ezplot(yzi,[0,8]);
grid on;
title('零输入响应');
subplot(3,1,2);ezplot(yzs,[0,8]);
grid on;
title('零状态响应');
subplot(3,1,3);ezplot(yt,[0,8]);
grid on;
title('完全响应');
```

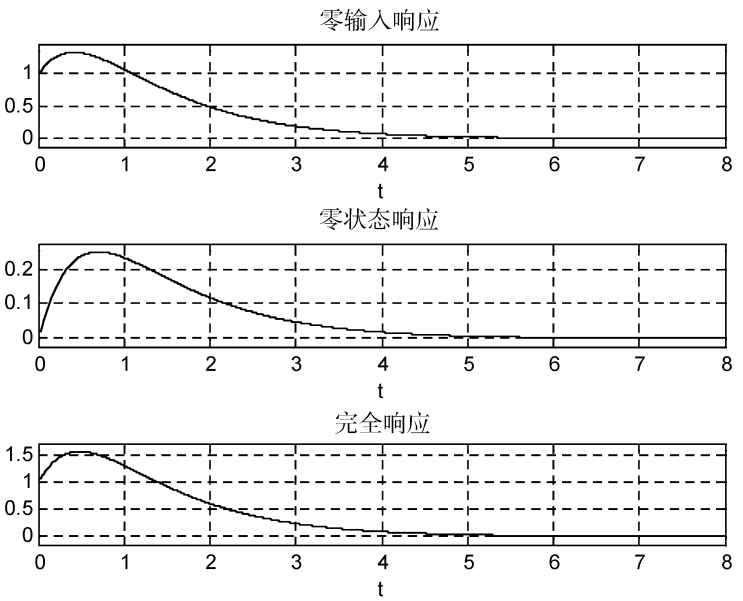


图 2-21　系统的响应

2.4.2　连续时间系统零状态响应的数值求解分析

前面叙述了符号求解系统微分方程的方法，实际工程中用得较多的方法是数值求解微分方程。下面主要讨论零状态响应的数值求解。而零输入响应的数值求解可通过函数 initial 来实现，

initial 函数中的参量必须是状态变量所描述的系统模型，此处不作说明。对于零状态响应，MATLAB 控制系统工具箱提供了对 LTI 系统的零状态响应进行数值仿真的函数 lsim，该函数可求解零初始条件下微分方程的数值解，其调用格式如下：

y=lsim(sys, f, t)

其中：b 和 a 分别为微分方程右端和左端的系数向量。例如，对于微分方程

$$a_3y'''(t)+a_2y''(t)+a_1y'(t)+a_0y(t)=b_3f'''(t)+b_2f''(t)+b_1f'(t)+b_0f(t)$$

可用 $\boldsymbol{a}$=[a_3,a_2,a_1,a_0]、$\boldsymbol{b}$=[b_3,b_2,b_1,b_0]、sys=*tf*($\boldsymbol{b}$, $\boldsymbol{a}$)获得其 LTI 模型。注意，如果微分方程的左端或右端表达式中有缺项，则其向量 $\boldsymbol{a}$ 或 $\boldsymbol{b}$ 中的对应元素应为 0，不能省略不写，否则会出错。

【例 2-21】已知某 LTI 系统的微分方程如下：

$$y''(t)+5y'(t)+6y(t)=6f(t)$$

式中：$f(t)=10\sin(2\pi t)u(t)$。试用 MATLAB 命令绘出 $0\leqslant t\leqslant 5$ 范围内系统零状态响应 $y(t)$的波形图。

其实现的 MATLAB 程序代码如下：

```
> clear all;
ts=0;te=5;
dt=0.01;
sys=tf([6],[1 5 6]);
t=ts:dt:te;
f=10*sin(2*pi*t).*uCT(t);
y=lsim(sys,f,t);
plot(t,y);
grid on;
xlabel('时间/s');ylabel('y(t)');
title('零状态响应');
```

运行程序，效果如图 2-22 所示。

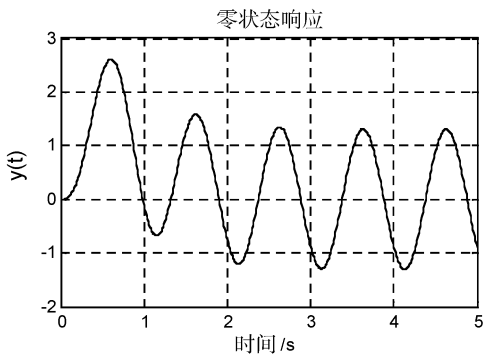


图 2-22　系统的零状态响应

【例 2-22】已知某 LTI 系统如下：

$$H(s)=\frac{\omega^2}{s^2+2s+\omega^2}\qquad(\omega=62.83)$$

求其系统的零状态响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
w2 = 62.83^2
h = tf(w2,[1 2 w2])
t = 0:0.1:5;
dt=0.016;
ts=0:dt:5;
us = (rem(ts,1)>=0.5);
hd = c2d(h,dt)
lsim(hd,us,ts)
xlabel('时间/s');ylabel('H(s)');
title('零状态响应');
```

运行程序，输出如下，效果如图 2-23 所示。

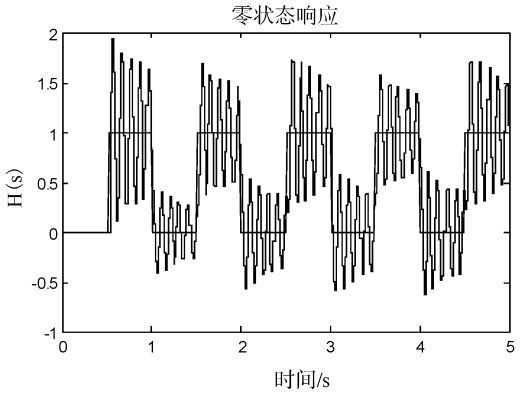


图 2-23　零状态响应效果

```
w2 =
   3.9476e+003
Transfer function:
         3948
----------------
s^2 + 2 s + 3948
Transfer function:
   0.4593 z + 0.4543
----------------------
z^2 - 1.055 z + 0.9685
Sampling time: 0.016
```

2.4.3　连续时间系统冲激响应和阶跃响应分析

在连续时间 LTI 系统中，冲激响应和阶跃响应是系统特性的描述，对它们的分析是线性系统中极为重要的问题。输入为单位冲激响应函数$\delta(t)$所引起的零状态响应称为单位冲激响应，简称冲激响应，用 $h(t)$表示；输入为单位阶跃函数 $u(t)$所引起的零状态响应称为单位阶跃响应，简称为阶跃响应，用 $g(t)$表示。

在 MATLAB 中，对于连续 LTI 系统的冲激响应和阶跃响应的数值解，可分别用控制系统工具箱提供的函数 impulse 和 step 来求解。其调用格式如下：

```
y=impulse(sys, t)
y=step(sys, t)
```

其中：t 表示计算系统响应的时间抽样点向量；sys 表示 LTI 系统模型。

【例 2-23】已知某 LTI 系统的微分方程如下：

$$y''(t)+2y'(t)+32y(t)=f'(t)+16f(t)$$

试用 MATLAB 命令绘出 $0\leqslant t\leqslant 4$ 范围内系统的冲激响应 $h(t)$和阶跃响应 $g(t)$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0:0.001:4;
sys=tf([1,16],[1,2,32]);
h=impulse(sys,t);     %冲激响应
g=step(sys,t);        %阶跃响应
subplot(2,1,1);plot(t,h);
grid on;
xlabel('时间/s');ylabel('h(t)');
title('冲激响应');
subplot(2,1,2);plot(t,g);
grid on;
xlabel('时间/s');ylabel('g(t)');
title('阶跃响应');
```

运行程序，效果如图 2-24 所示。

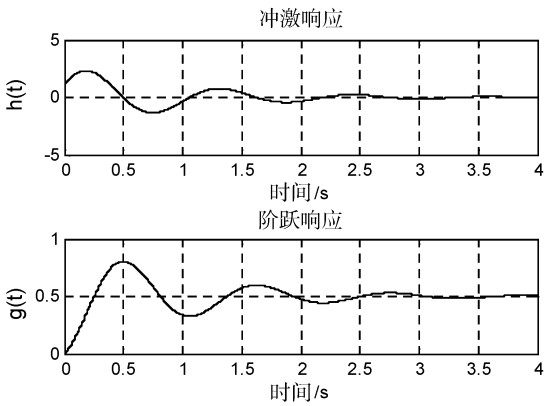


图 2-24　冲激响应和阶跃响应

2.4.4　利用卷积积分法求系统的零状态响应

由卷积积分公式可以得出，LTI 系统对于任意输入信号的零状态响应，可由系统的单位冲激响应与输入信号的卷积积分得到。卷积积分提供了求系统零状态响应的另一途径，利用 MATLAB 可以方便计算。卷积积分还是联系时域和频域的基本概念，建立了信号与系统的时域和频域之间的关系，同时将系统分析的时域方法、傅里叶变换方法和拉普拉斯变换统一起来。

【例 2-24】已知某 LTI 系统的微分方程如下：

$$y''(t)+2y'(t)+32y(t)=f'(t)+16f(t)$$

式中：$f(t)=e^{-2t}$。试利用 MATLAB 卷积积分法绘出系统零状态响应 $y(t)$的波形图。

利用卷积积分法求解。从例 2-23 中可以看出，系统的冲激响应 $h(t)$并不是时限信号，且激励信号 $f(t)$也不是时限信号，但可设置一定的时间范围使 $f(t)$和 $h(t)$衰减到足够小，从而近似地

求出零状态响应。取 t=[0,4]。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
dt=0.01; t1=0:dt:4;
f1=exp(-2*t1);
t2=t1;
sys=tf([1,16],[1,2,32]);
f2=impulse(sys,t2);
[t,f]=ctsconv(f1,f2,t1,t2,dt);
```

运行程序，效果如图 2-25 所示。

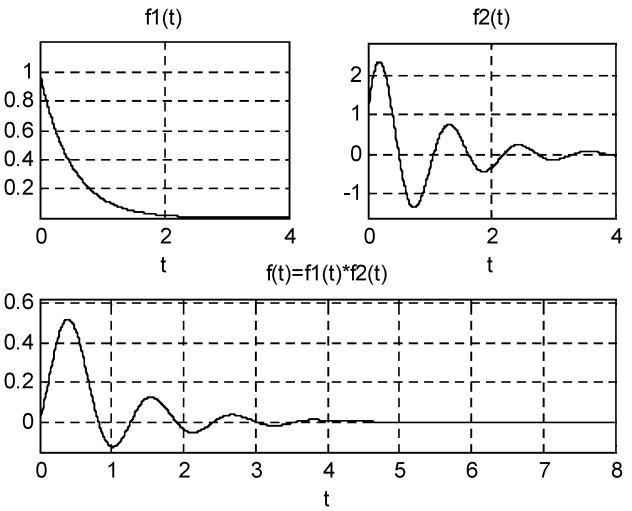


图 2-25　卷积积分求解零状态响应

在运行程序过程中调用到用户自定义编写的连续时间信号卷积运算函数 ctsconv.m。其源代码如下：

```
function   [f,t]=ctsconv(f1,f2,t1,t2,dt)
f=conv(f1,f2);
f=f*dt;
ts=min(t1)+min(t2);
te=max(t1)+max(t2);
t=ts:dt:te;
subplot(2,2,1);plot(t1,f1);
grid on;
axis([min(t1),max(t1),min(f1)-abs(min(f1)*0.2),max(f1)+abs(max(f1)*0.2)]);
title('f1(t)');xlabel('t');
subplot(2,2,2);plot(t2,f2);
grid on;
axis([min(t2),max(t2),min(f2)-abs(min(f2)*0.2),max(f2)+abs(max(f2)*0.2)]);
title('f2(t)');xlabel('t');
subplot(2,1,2);plot(t,f);
grid on;
axis([min(t),max(t),min(f)-abs(min(f)*0.2),max(f)+abs(max(f)*0.2)]);
title('f(t)=f1(t)*f2(t)');xlabel('t');
```

也可以用前面提到的 lsim 函数来求解，其实现的 MATLAB 程序代码如下：

```
>> clear all;
ts=0;te=4;
dt=0.01;
sys=tf([1,16],[1,2,32]);
t=ts:dt:te;
f=exp(-2*t);
y=lsim(sys,f,t);
plot(t,y);grid on;
xlabel('时间/s');ylabel('y(t)');
title('零状态响应');
```

运行程序，结果如图 2-26 所示。与图 2-25 比较可知，通过卷积积分法计算零状态响应，其结果与直接计算结果相同。

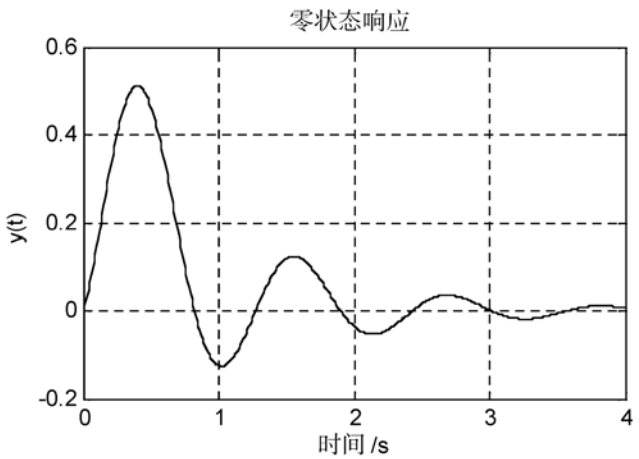


图 2-26　直接数值解法求解零状态响应

2.5　离散时间信号在 MATLAB 中的运算

2.5.1　离散时间信号的基本运算

对离散时间序列实行基本运算可得到新的序列，这些基本运算主要包括加、减、乘、除、移位和反折等。两个序列的加减乘除是对应离散样点值的加减乘除，因此，可通过 MATLAB 的点乘和点除、序列移位和反折来实现，与连续时间信号处理方法基本一样。

【例 2-25】用 MATLAB 命令画出下列离散时间信号的波形图。

（1）$x_1(n)=a^n[u(n)-u(n-N)]$　　（2）$x_2(n)=x_1(n+3)$

（3）$x_3(n)=x_1(n-2)$　　（4）$x_4(n)=x_1(-n)$

设 a=0.8，N=8，其实现的 MATLAB 程序代码如下：

```
>> clear all;
a=0.8;N=8;
n=-12:1:12;
x=a.^n.*(uDT(n)-uDT(n-N));
```

```
n1=n;n2=n1+3;
n3=n1-2;n4=-n1;
subplot(4,1,1);stem(n1,x,'fill');
grid on;
axis([-15 15 0 1]);
xlabel('(a) x1(n)');
subplot(4,1,2);stem(n2,x,'fill');
grid on;
axis([-15 15 0 1]);
xlabel('(b) x2(n)');
subplot(4,1,3);stem(n3,x,'fill');
grid on;
axis([-15 15 0 1]);
xlabel('(c) x3(n)');
subplot(4,1,4);stem(n4,x,'fill');
grid on;
axis([-15 15 0 1]);
xlabel('(d) x4(n)');
```

运行程序，输出如图 2-27 所示。

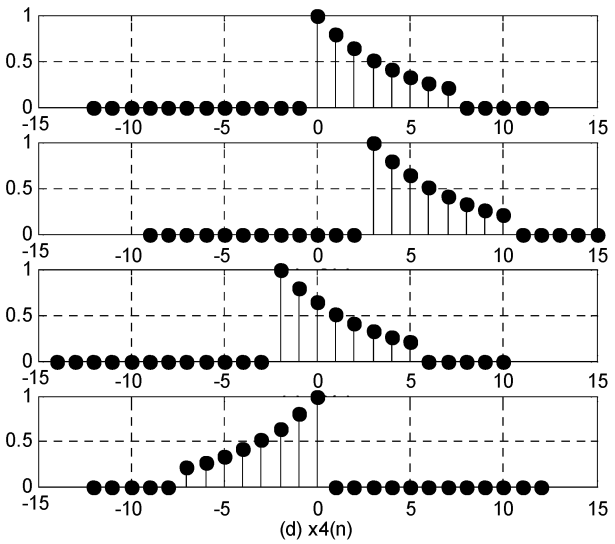


图 2-27　离散时间信号的基本运算及波形图

在运行程序过程中，调用到自定义的冲激序列 uDT.m 文件。其源代码如下：

```
function y=uDT(n)
y=n>=0;            %当参数为非负时输出 1
```

2.5.2　离散时间系统的响应

离散时间 LTI 系统可用线性常系数差分方程来描述，即

$$\sum_{i=0}^{N} a_i y(n-i) = \sum_{j=0}^{M} b_j x(n-j) \qquad (2\text{-}4)$$

式中：$a_i(i=0,1,\cdots,N)$和 $b_j(0,1,\cdots,M)$为实常数。

MATLAB 中的函数 filter 可对式（2-4）的差分方程在指定时间范围内的输入序列所产生的响应进行求解。函数 filter 的调用格式如下：

y=filter(b, a, x)

其中：x 为输入的离散序列；y 为输出的离散序列；y 的长度与 x 的长度一样；b 与 a 分别为差分方程右端与左端的系数向量。

【例 2-26】已知某 LTI 系统的差分方程如下：

$$3y(n)-4y(n-1)+2y(n-2)=x(n)+2x(n-1)$$

试用 MATLAB 命令绘出当激励信号为 $x(n)=\left(\frac{1}{2}\right)^n u(n)$ 时，该系统的零状态响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a=[3 -4 2];
b=[1 2];
n=0:30;
x=(1/2).^n;
y=filter(b,a,x);
stem(n,y,'fill');
grid on;
xlabel('n'); title('系统零状态响应 y(n)');
```

运行程序，效果如图 2-28 所示。

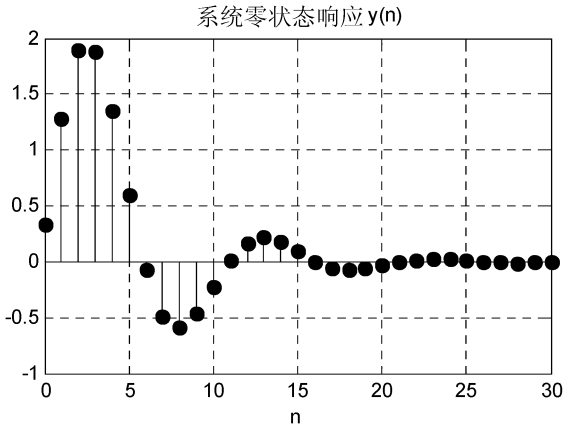


图 2-28　系统的零状态响应

2.5.3　离散时间系统的单位取样响应

系统的单位取样响应定义为系统在$\delta(n)$激励下系统的零状态响应，用 $h(n)$表示。MATLAB 求解单位取样响应可利用函数 filter，并将激励设为如下所定义的 impDT 函数。例如，求解例 2-20 中系统的单位取样响应时，MATLAB 源程序如下：

```
>> clear all;
a=[3 -4 2];
```

```
b=[1 2];
n=0:30;
x=impDT(n);
y=filter(b,a,x);
stem(n,y,'fill');
grid on;
xlabel('n'); title('系统零状态响应 y(n)');
```

运行程序，效果如图 2-29 所示。

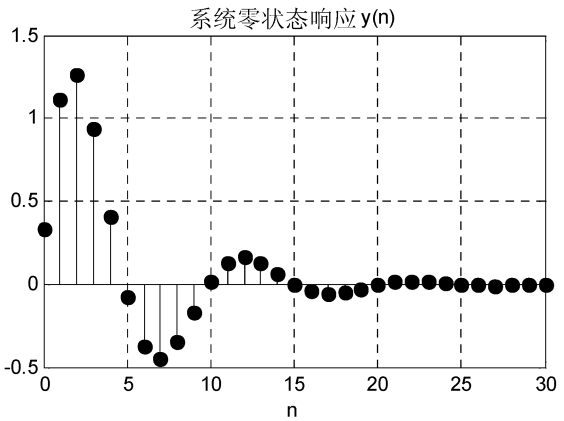


图 2-29　系统单位取样响应

impDT 函数的源程序代码如下：

```
function y=impDT(n)
y=(n= =0);     %当参数为 0 时冲激为 1,否则为 0
```

MATLAB 的另一种求单位取样响应的方法是利用控制系统工具箱提供的函数 impz 来实现。impz 函数的常用语句格式如下：

```
[h,t] = impz(ha)
[h,t] = impz(...,fs)
impz(b, a, N)
[h,t] = impz(hd)
impz(hd)
[h,t] = impz(hm)
impz(hm)
```

【例 2-27】impz 函数用法示例。

运行程序，效果如图 2-30 所示。

```
>> d = fdesign.lowpass(.4,.5,1,80)
d =
                    Response: 'Lowpass'
               Specification: 'Fp,Fst,Ap,Ast'
                 Description: {4x1 cell}
      NormalizedFrequency: true
                       Fpass: 0.4
                       Fstop: 0.5
```

```
                    Apass: 1
                    Astop: 80
>> hd=design(d,'ellip')
hd =
          FilterStructure: 'Direct-Form II, Second-Order Sections'
               Arithmetic: 'double'
                sosMatrix: [4x6 double]
              ScaleValues: [5x1 double]
    OptimizeScaleValues: true
        PersistentMemory: false
>> impz(hd)
```

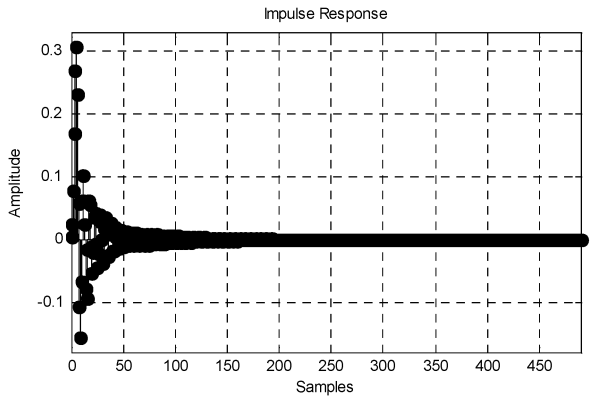


图 2-30　impz 函数显示效果

【例 2-28】已知某 LTI 系统的差分方程如下：

$$3y(n)-4y(n-1)+2y(n-2)=x(n)+2x(n-1)$$

利用 MATLAB 的 impz 函数绘出该系统的单位取样响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a=[3 -4 2];
b=[1 2];
n=0:30;
impz(b,a,30);
grid on;
title('系统单位取样响应 h(n)');
```

运行程序，效果如图 2-31 所示。

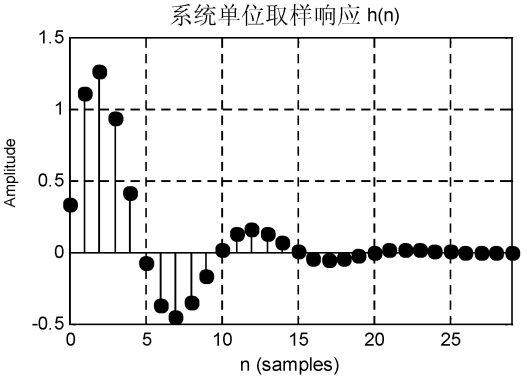


图 2-31　系统单位取样响应

2.5.4　离散时间信号的卷积和运算

由于系统的零状态响应是激励与系统的单位取样响应的卷积，因此，卷积运算在离散时间信号处理领域被广泛应用。离散时间信号的卷积定义如下：

$$y(n) = x(n) * h(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m) \qquad (2\text{-}5)$$

可见，离散时间信号的卷积运算是求和运算，因此常称为“卷积和”。

MATLAB 求离散时间信号卷积和的命令为 conv，其调用格式如下：

```
y=conv(x,h)
```

其中：x 与 h 表示离散时间信号值的向量；y 为卷积结果。用 MATLAB 进行卷积和运算时，无法实现无限的累加，只能计算时限信号的卷积。

例如，利用 MATLAB 的 conv 命令求两个长为 4 的矩形序列的卷积和，即 $g(n)=[u(n)-u(n-4)]*[u(n)-u(n-4)]$，其结果应是长为 7（4+4−1=7）的三角序列。用向量[1 1 1 1]表示矩形序列，MATLAB 源程序如下：

```
>> x1=[1 1 1 1];
x2=[1 1 1 1];
g=conv(x1,x2)
g =
     1     2     3     4     3     2     1
```

如果要绘出图形，则利用 stem 命令，即

```
>> n=1:7;
stem(n,g,'fill');
grid on;
xlabel('n');
```

运行程序，输出效果如图 2-32 所示。

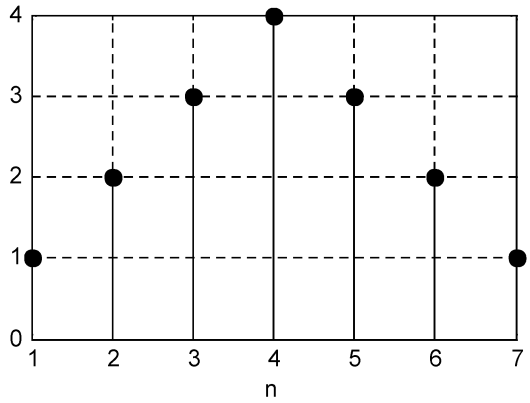


图 2-32　卷积结果图

对于给定函数的卷积和，应计算卷积结果的起始点及其长度。两个时限序列的卷积和长度一般等于两个序列长度的和减 1。

【例 2-29】已知某系统的单位取样响应为 $h(n)=0.8^n[u(n)-u(n-8)]$，试用 MATLAB 求当激励信号为 $x(n)=u(n)-u(n-4)$时，系统的零状态响应。

在 MATLAB 中可通过卷积求解零状态响应，即 $x(n)*h(n)$。由题意可知，描述 $h(n)$向量的长度至少为 8，描述 $x(n)$向量的长度至少为 4，因此为了图形完整美观，将 $h(n)$向量和 $x(n)$向量加上一些附加的零值。

其实现的 MATLAB 的程序代码如下：

```
>> clear all;
nx=-1:5;        %x(n)向量显示范围(添加了附加的零值)
nh=-2:10;       %h(n)向量显示范围(添加了附加的零值)
x=uDT(nx)-uDT(nx-4);
h=0.8.^nh.*(uDT(nh)-uDT(nh-8));
y=conv(x,h);
ny1=nx(1)+nh(1);     %卷积结果起始点
% 卷积结果长度为两序列长度之和减 1,即 0 到(length(nx)+length(nh)-2)
% 因此卷积结果的时间范围是将上述长度加上起始点的偏移值
ny=ny1+(0:(length(nx)+length(nh)-2));
subplot(3,1,1);stem(nx,x,'fill');
grid on;
xlabel('n');title('x(n)');
axis([-4 16 0 3]);
subplot(3,1,2);stem(nh,h,'fill');
grid on;
xlabel('n');title('h(n)');
axis([-4 16 0 3]);
subplot(3,1,3);stem(ny,y,'fill');
grid on;
xlabel('n');title('y(n)=x(n)*h(n)');
axis([-4 16 0 3]);
```

运行程序，效果如图 2-33 所示。

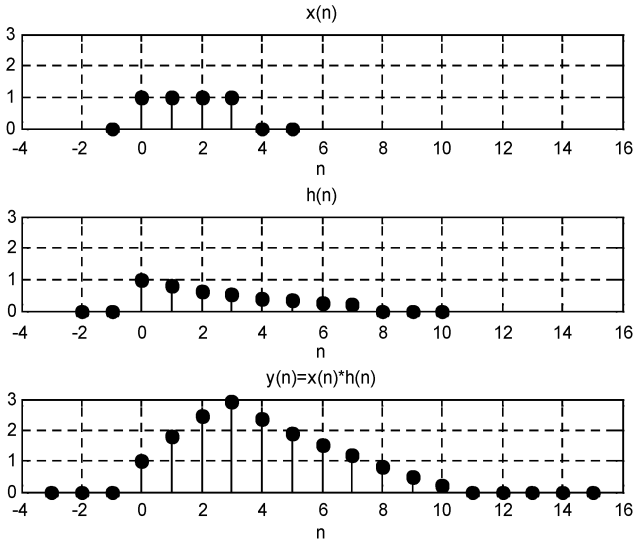


图 2-33　利用卷积和法求解系统的零状态响应

2.6　信号抽样及抽样定理

2.6.1　信号抽样分析

信号抽样是连续时间信号分析向离散时间信号分析、连续信号处理向数字信号处理的第一步，广泛应用于实际的各类系统中。信号抽样也称为取样或采样，就是利用抽样脉冲序列 $p(t)$ 从连续信号 $f(t)$中抽取一系列的离散样值，通过抽样过程得到的离散样值信号称为抽样信号，用 $f_s(t)$表示。从数学上讲，抽样过程就是抽样脉冲 $p(t)$和原连续信号 $f(t)$相乘的过程，即

$$f_s(t) = f(t)p(t)$$

因此，可以用傅里叶变换的频域卷积性质来求抽样信号 $f_s(t)$的频谱。常用的抽样脉冲序列 $p(t)$有周期矩形脉冲序列和周期冲激脉冲序列。

假设原连续信号 $f(t)$的频谱为 $F(\omega)$，即 $f(t) \leftrightarrow F(\omega)$；抽样脉冲 $p(t)$是一个周期信号，它的频谱为

$$p(t) = \sum_{n=-\infty}^{\infty} P_n \mathrm{e}^{\mathrm{j}n\omega_s t} \leftrightarrow P(\omega) = 2\pi \sum_{n=-\infty}^{\infty} P_n \delta(\omega - n\omega_s)$$

式中：$\omega_s = \dfrac{2\pi}{T_s}$ 为抽样角频率；T_s 为抽样间隔。因此，抽样信号 $f_s(t)$的频谱为

$$F_s(\omega) = \frac{1}{2\pi} F(\omega)P(\omega) = \sum_{n=-\infty}^{\infty} F(\omega)P_n\delta(\omega - n\omega_s) = \sum_{n=-\infty}^{\infty} P_n F(\omega - n\omega_s)$$

即

$$F_s(\omega) = \sum_{n=-\infty}^{\infty} P_n F(\omega - n\omega_s) \qquad (2\text{-}6)$$

式（2-6）表明，信号在时域被抽样后，它的频谱是原连续信号的频谱以抽样和角频率为间

隔周期的延拓，即信号在时域抽样或离散化，相当于频域周期化。在频谱的周期重复过程中，其频谱幅度受抽样脉冲序列的傅里叶系数加权，即被 P_n 加权。

假设抽样信号为周期冲激序列，则

$$p(t)=\sum_{n=-\infty}^{\infty}\delta(t-nT_s)\leftrightarrow\omega_s\sum_{n=-\infty}^{\infty}\delta(\omega-n\omega_s)$$

因此，冲激脉冲序列抽样后信号的频谱为

$$F_s(\omega)=\frac{1}{T_s}\sum_{n=-\infty}^{\infty}F(\omega-n\omega_s)$$

可以看出，$F_s(\omega)$是以ω_s为周期等幅地重复。

【例 2-30】已知升余弦脉冲信号为

$$f(t)=\frac{F}{2}\left[1+\cos\left(\frac{\pi t}{\tau}\right)\right]\quad(0\leqslant t\leqslant\tau)$$

用 MATLAB 编程实现该信号经冲激脉冲抽样后得到的抽样信号 $f_s(t)$及其频谱。

参数 E=1、τ=π，则 $f(t)=\frac{1}{2}(1+\cos t)$。当采样抽样间隔 T_s=1 时，其实现的 MATLAB 程序代码如下：

```
>> clear all;
Ts=1;              %抽样间隔
dt=0.1;
t1=-4:dt:4;
ft=((1+cos(t1))/2).*(uCT(t1+pi)-uCT(t1-pi));
subplot(2,2,1);plot(t1,ft);
grid on;
axis([-4 4 -0.1 1.1]);
xlabel('Time(sec)');ylabel('f(t)');
title('升余弦脉冲信号');
N=500;
k=-N:1:N;
W=pi*k/(N*dt);
Fw=dt*ft*exp(-j*t1'*W);   %傅里叶变换的数值计算
subplot(2,2,2);plot(W,abs(Fw));
grid on;
axis([-10 10 -0.2 1.1*pi]);
xlabel('\omega');ylabel('F(w)');
title('升余弦脉冲信号的频谱');
t2=-4:Ts:4;
fst=((1+cos(t2))/2).*(uCT(t2+pi)-uCT(t2-pi));
subplot(2,2,3);plot(t1,ft,':');   %抽样信号的包络线
hold on;
stem(t2,fst);     %绘制抽样信号
grid on;
axis([-4 4 -0.1 1.1]);
```

```
xlabel('Time(sec)');ylabel('fs(t)');
title('抽样后的信号');
hold off;
Fsw=Ts*fst*exp(-j*t2'*W);   %傅里叶变换的数值计算
subplot(2,2,4);plot(W,abs(Fsw));
grid on;
axis([-10 10 -0.2 1.1*pi]);
xlabel('\omega');ylabel('Fs(w)');
title('抽样信号的频谱');
```

运行程序，效果如图 2-34 所示。

很明显，升余弦脉冲信号的频谱在抽样后发生了周期延拓，频域上该周期为$\omega_s=2\pi/T_s$。

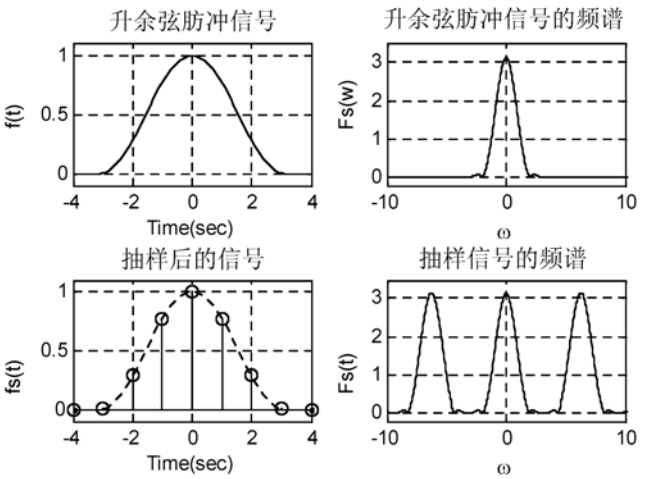


图 2-34　升余弦脉冲信号经抽样后的频谱比较

2.6.2　抽样定理分析

若 $f(t)$是带限信号，带宽为ω_m，则信号 $f(t)$可以用等间隔的抽样值来唯一表示。$f(t)$经抽样后的频谱 $F_s(\omega)$就是将 $f(t)$的频谱 $F(\omega)$在频谱轴上以抽样频率ω_s为间隔进行周期延拓。因此，当$\omega_s \geqslant 2\omega_m$时，或者抽样间隔 $T_s \leqslant \dfrac{\pi}{\omega_m}\left(T_s=\dfrac{2\pi}{\omega_s}\right)$时，周期延拓后频谱 $F_s(\omega)$不会产生频率混叠；当$\omega_s<2\omega_m$时，周期延拓后频谱 $F_s(\omega)$将产生频率混叠。通常把满足抽样定理要求的最低抽样频率 $f_s=2f_m\left(f_s=\dfrac{\omega_s}{2\pi}, f_m=\dfrac{\omega_m}{2\pi}\right)$称为奈奎斯特频率，把最大允许的抽样间隔$T_s=\dfrac{1}{f_s}=\dfrac{1}{2f_m}$称为奈奎斯特间隔。

【例 2-31】试用例 2-30 来验证抽样定理。

例 2-30 中升余弦脉冲信号的频谱大部分集中在$\left[0,\dfrac{2\pi}{\tau}\right]$，设其截止频率为$\omega_m=\dfrac{2\pi}{\tau}$，代入参

数可得$\omega_m 2$，因而奈奎斯特间隔$T_s=\dfrac{1}{2f_m}=\dfrac{\pi}{2}$。在例 2-30 的 MATLAB 程序中，可通过修改 T_s 的值得到不同的结果。

例如，取 T_s=pi/2，可得到奈奎斯特间隔临界抽样时，抽样信号的频谱情况，如图 2-35 所示。取 T_s=2，可得到低抽样率时，抽样信号的频谱情况，如图 2-36 所示。从中可以看出，由于抽样间隔大于奈奎斯特间隔，产生了较为严重的频谱混叠现象。

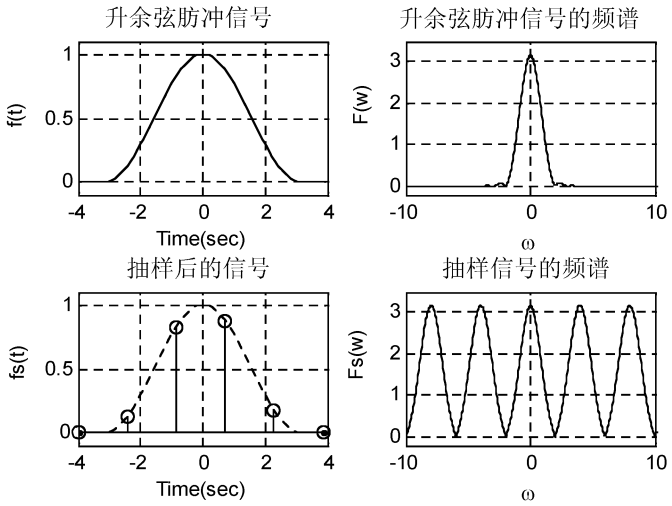


图 2-35　临界抽样时抽样信号频谱比较

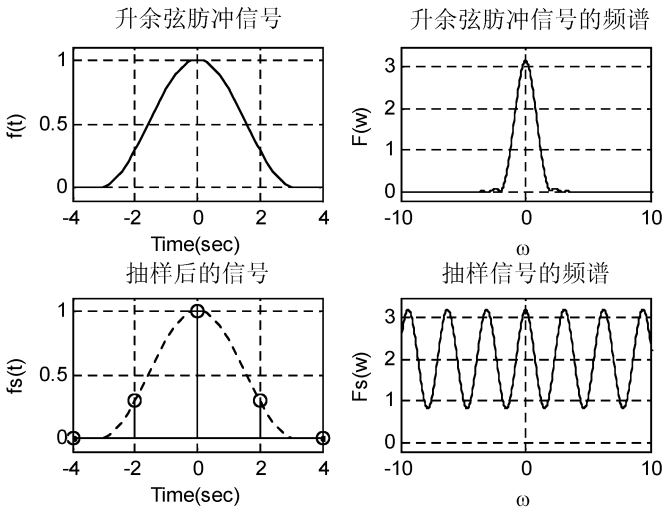


图 2-36　低抽样率时抽样信号频谱比较及频率混叠

2.6.3　信号重建分析

抽样定理表明，当抽样间隔小于奈奎斯特间隔时，可以用抽样信号 $f_s(t)$唯一地表示原信号 $f(t)$，即信号的重建。为了从频谱中无失真地恢复原信号，可采用截止频率为$\omega_c \geqslant \omega_m$的理想低通滤波器。

设理想低通滤波器的冲激响应为 $h(t)$，即

$$f(t)=f_s(t)*h(t)$$

其中：$f_s(t)=f(t)\sum_{n=-\infty}^{\infty}\delta(t-nT_s)=\sum_{n=-\infty}^{\infty}f(nT_s)\delta(t-nT_s)$，$h(t)=T_s\frac{\omega_c}{\pi}\mathrm{Sa}(\omega_c t)$，则有

$$f(t)=\sum_{n=-\infty}^{\infty}f(nT_s)\delta(t-nT_s)T_s\frac{\omega_c}{\pi}\mathrm{Sa}(\omega_c t)=T_s\frac{\omega_c}{\pi}\sum_{n=-\infty}^{\infty}f(nT_s)\mathrm{Sa}[\omega_c(t-nT_s)] \qquad (2\text{-}7)$$

式（2-7）表明，连续信号可以展开为抽样函数 Sa(*t*)的无穷级数，该级数的系数等于抽样值。

利用 MATLAB 中的函数 $\sin c(t)\frac{\sin(\pi t)}{\pi t}$ 来表示 Sa(*t*)，有 $\mathrm{Sa}(t)=\sin c\left(\frac{t}{\pi}\right)$，所以可获得由 $f(nT_s)$ 重建 *f*(*t*)的表达式，即

$$f(t)=T_s\frac{\omega_c}{\pi}\sum_{n=-\infty}^{\infty}f(nT_s)\sin c\left[\frac{\omega_c}{\pi}(t-nT_s)\right] \qquad (2\text{-}8)$$

【例 2-32】对例 2-30 中的升余弦脉冲信号，假设其截止频率ω_m=2，抽样间隔 T_s=1，采样截止频率ω_c=1.2×ω_m 的低通滤波器对抽样信号滤波后重建信号 *f*(*t*)，并计算重建信号与原升余弦脉冲信号的绝对误差。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wn=2;                                          %升余弦脉冲信号带宽
wc=1.2*wn;                                     %理想低通截止频率
Ts=1;                                          %抽样间隔
n=-100:1:100;                                  %时域计算点数
nTs=n*Ts;                                      %时域抽样点
fs=((1+cos(nTs))/2).*(uCT(nTs+pi)-uCT(nTs-pi)); %抽样信号
t=-4:0.1:4;
ft=fs*Ts*wc/pi*sinc((wc/pi)*(ones(length(nTs),1)*t-nTs'*ones(1,length(t))));
t1=-4:0.1:4;
f1=((1+cos(t1))/2).*(uCT(t1+pi)-uCT(t1-pi));
subplot(3,1,1);plot(t1,f1,':');                %创建包络线
hold on;
stem(nTs,fs);                                  %绘制抽样信号
grid on;
axis([-4 4 -0.1 1.1]);
xlabel('nTs');ylabel('f(nTs)');
title('抽样间隔 Ts=1 时的抽样信号 f(nTs)');
hold off
subplot(3,1,2);plot(t,ft);                     %绘制重建信号
grid on;
axis([-4 4 -0.1 1.1]);
xlabel('t');ylabel('f(t)');
title('由 f(nTs)信号重建得到升余弦脉冲信号');
error=abs(ft-f1);
subplot(3,1,3);plot(t,error);
grid on;
xlabel('t');ylabel('error(t)');
```

```
title('重建信号与原升余弦脉冲信号的绝对误差');
```

运行程序，效果如图 2-37 所示。从图 2-37 中可知，重建后的信号与原升余弦脉冲信号的误差在 10^2 以内，这是因为当选取升余弦脉冲信号带宽 ω_m=2 时，实际上已经将很少的高频分量忽略了。

【例 2-33】如果将例 2-32 中的抽样间隔修改为 T_s=2，低通滤波器的截止频率修改为 $\omega_c=\omega_m$，那么，按照例 2-31 的分析将会产生频率混叠，则重建的信号与原来的升余弦脉冲信号相比也会产生较大失真。按要求修改上述 MATLAB 的程序，并分析失真的误差。

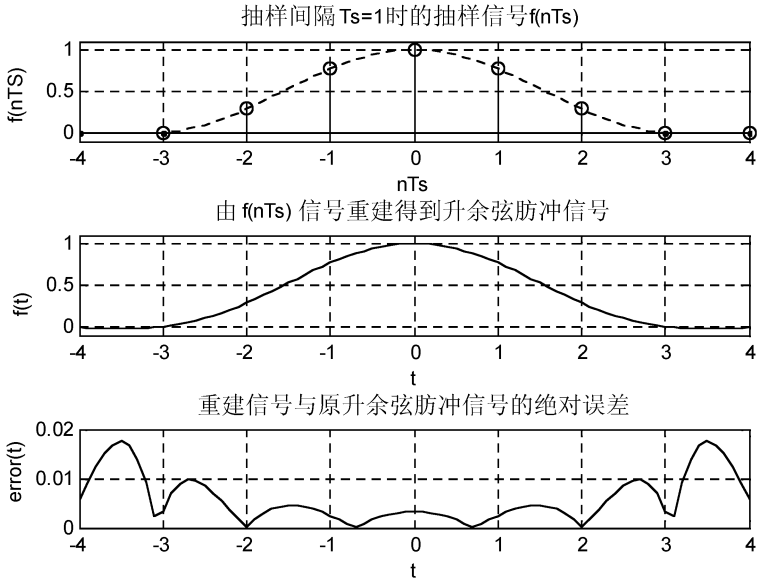


图 2-37　抽样信号的重建及误差分析

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wn=2;                                          %升余弦脉冲信号带宽
wc=wn;                                         %理想低通截止频率
Ts=2;                                          %抽样间隔
n=-100:1:100;                                  %时域计算点数
nTs=n*Ts;                                      %时域抽样点
fs=((1+cos(nTs))/2).*(uCT(nTs+pi)-uCT(nTs-pi)); %抽样信号
t=-4:0.1:4;
ft=fs*Ts*wc/pi*sinc((wc/pi)*(ones(length(nTs),1)*t-nTs'*ones(1,length(t))));
t1=-4:0.1:4;
f1=((1+cos(t1))/2).*(uCT(t1+pi)-uCT(t1-pi));
subplot(3,1,1);plot(t1,f1,':');                %创建包络线
hold on;
stem(nTs,fs);                                  %绘制抽样信号
grid on;
axis([-4 4 -0.1 1.1]);
xlabel('nTs');ylabel('f(nTs)');
title('抽样间隔 Ts=2 时的抽样信号 f(nTs)');
hold off
```

```
subplot(3,1,2);plot(t,ft);                          %绘制重建信号
grid on;
axis([-4 4 -0.1 1.1]);
xlabel('t');ylabel('f(t)');
title('由 f(nTs)信号重建得到有失真的升余弦脉冲信号');
error=abs(ft-f1);
subplot(3,1,3);plot(t,error);
grid on;
xlabel('t');ylabel('error(t)');
title('重建信号与原升余弦脉冲信号的绝对误差');
```

运行程序，效果如图 2-38 所示。

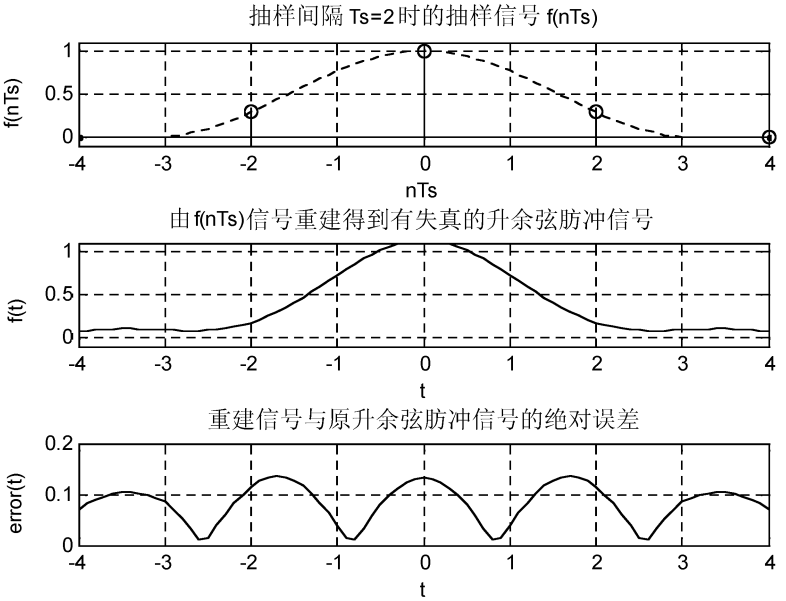


图 2-38　不满足抽样定理条件的信号的重建

图 2-38 反映了不满足抽样定理时，即抽样间隔大于奈奎斯特间隔的情况下信号的重建。与升余弦脉冲信号作比较可发现有较大的失真产生，且绝对误差十分明显。

第 3 章　系统模型及数据采集分析

3.1　系统数学模型

1．传递函数

传递函数是离散系统（如数字滤波器）的基本 z 域表示形式，它是两个多项式之比。离散系统传递函数的 Z 变换形式如下：

$$Y(z)=\frac{b(1)+b(2)z^{-1}+\cdots+b(nb+1)z^{-nb}}{1+a(2)z^{-1}+\cdots+a(na+1)z^{-na}}X(z) \tag{3-1}$$

其中：$b(i)$与 $a(i)$是系统传递函数的系数，系统阶次是 na 和 nb 中的最大值，系统传递函数的系数分别储存在两个向量（一般是行向量）中。

MATLAB 中用系统传递函数的分子和分母的系数构成的两个向量来唯一确定一个系统，即

$$\text{num}=[a(1),a(2),\cdots,a(n)]$$
$$\text{den}=[b(1),b(2),\cdots,b(n)]$$

2．状态空间

状态方程是描述系统的一种常用方式，这种方式基于系统的不可见的状态变量，所以，又称为系统的内部描述方法，即

$$\boldsymbol{x}(n+1)=\boldsymbol{A}\boldsymbol{x}(n)+\boldsymbol{B}\boldsymbol{u}(n)$$
$$\boldsymbol{y}(n)=\boldsymbol{C}\boldsymbol{x}(n)+\boldsymbol{D}\boldsymbol{u}(n)$$

其中：$\boldsymbol{u}(n)$是输入向量；$\boldsymbol{x}$ 是状态向量；$\boldsymbol{y}$ 是输出向量；$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 分别为常数矩阵。在 MATLAB 中，一般情况下，系统的状态方程可以简记为（$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$），如果 $\boldsymbol{D}$=0，则系统的状态方程模型可以简记为（$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$）。

3．零极点增益

零极点增益实际上是传递函数模型的另一种表现形式，其原理分别是对原系统传递函数的分子和分母进行分解因式处理，以获得系统的零极点表示形式，对单输入单输出系统来说，可以简单地将其零极点模型写为

$$H(z)=\frac{q(z)}{p(z)}=k\frac{(z-q(1))(z-q(2))\cdots(z-q(n))}{(z-p(1))(z-p(2))\cdots(z-p(n))}$$

其中：$q(i)$，i=1,2,⋯,m 和 $p(i)$，i=1,2,⋯,n 分别称为系统的零点和极点，它们既可以为实数也可以为复数，而 k 称为系统的增益。在 MATLAB 中滤波器就可以简记为［Z，P，K］。

4．二阶分割形式

离散传递函数的二阶分割形式如下：

$$H(z)=\prod_{k=1}^{L}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

sos 是二阶分割形式的系数矩阵，即

$$\boldsymbol{sos}=\begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \end{bmatrix}$$

系统传递函数是二阶分割形式传递函数的增益 G 次阶，如果 G 没有给定，则默认为 1。

5．部分分式形式

每一个传递函数都有对应的部分分式或留数，形式如下：

$$\frac{b(z)}{a(z)}=\frac{r(1)}{1-p(1)z^{-1}}+\cdots+\frac{r(n)}{1-p(n)z^{-1}}$$

给出的 $H(z)$没有重极点。此外 n 是有理传递函数 $b(z)/a(z)$分母多项式的次数。若 r 表示 S_r 的重复次数，则 $H(z)$具有如下形式：

$$\frac{r(j)}{1-p(j)z^{-1}}+\frac{r(j+1)}{(1-p(j)z^{-1})^2}+\cdots+\frac{r(j+S_r-1)}{(1-p(j)z^{-1})^{S_r}}$$

在 MATLAB 中提供了相关函数实现以上各系统的模型。下面介绍如下：

（1）conv 函数。功能：卷积与多项式乘积。其调用格式如下：

c=conv(a,b)：计算向量 A 与 B 的卷积，卷积的长度为 length(A)+length(B)−1，如果 A 与 B 是多项式的系数，则卷积的结果是它们乘积多项式的系数。conv 函数允许进行多级嵌套使用。

【例 3-1】运用 conv 表示传递函数：

$$G(s)=\frac{25}{(s+6)(s^2+105s+75)(s^3+34s^2+50s+10)}$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
num=[25];
den=conv([1,6],conv([1,105,75],[1,34,50,10]))
```

（2）convmtx 函数。功能：卷积矩阵。其调用格式如下：

A = convmtx(c,n)

A = convmtx(r,n)

返回向量 c 的卷积矩阵，如果 c 是一个行向量，x 是一长度为 n 的行向量，则 convmtx(c, n)*x 等同于 conv(c, x)；如果 r 是列向量，x 是一个长度为 c 的行向量，则 x*convmtx(r, n)等同于 conv(r,x)。

【例 3-2】convmtx 用法。

```
>> h = [1 2 3 2 1];
convmtx(h,7)
```

运行程序，输出如下：

```
ans =
     1     2     3     2     1     0     0     0     0     0     0
     0     1     2     3     2     1     0     0     0     0     0
     0     0     1     2     3     2     1     0     0     0     0
     0     0     0     1     2     3     2     1     0     0     0
     0     0     0     0     1     2     3     2     1     0     0
     0     0     0     0     0     1     2     3     2     1     0
```

```
     0     0     0     0     0     0     1     2     3     2     1
```

（3）latc2tf 函数。功能：将格式滤波器形式转变为传递函数形式。其调用格式如下：

[num,den] = latc2tf(k,v)：由 IIR 滤波器的格式系数 k 和梯形系数 v 得到其传递函数的分子系数 num 和分母系数 den。

[num,den] = latc2tf(k,'iiroption')：由全极点 IIR 滤波器形式系数 k 得到其传递函数的分子系数 num 和分母系数。

num = latc2tf(k,'firoption')：由 FIR 滤波器格式系数 k 得到其传递函数分子系数 num。

（4）sos2ss 函数。功能：将二阶分割形式转换为状态方程形式。其调用格式如下：

[A,B,C,D] = sos2ss(sos)

[A,B,C,D] = sos2ss(sos,g)

由二阶形式的增益 g 与 sos 矩阵得到状态方程矩阵 a, b, c 和 d。

【例 3-3】sos2ss 用法。

```
>> sos = [1   1   1   1   0 -1; -2   3   1   1 10   1];
[A,B,C,D] = sos2ss(sos)
```

运行程序，输出如下：

```
A =
    -10     0     10     1
     1      0      0     0
     0      1      0     0
     0      0      1     0
B =
     1
     0
     0
     0
C =
    21     2    -16    -1
D =
    -2
```

（5）sos2tf 函数。功能：将二阶分割形式转变为传递函数形式。其调用格式如下：

[b,a] = sos2tf(sos)

[b,a] = sos2tf(sos,g)

由二阶分割形式的系数矩阵 sos 与增益 g 得出离散线性系统的传递函数的分子与分母系数 b 和 a。

【例 3-4】sos2tf 用法。

```
>> sos = [1   1   1   1   0 -1; -2   3   1   1 10   1];
[b,a] = sos2tf(sos)
```

运行程序，输出如下：

```
b =
    -2     1     2     4     1
a =
     1    10     0   -10    -1
```

（6）sos2zp 函数。功能：将二阶分割形式转变为零极点增益形式。其调用格式如下：

```
[z,p,k] = sos2zp(sos)
[z,p,k] = sos2zp(sos,g)
```

由二阶分割形式给出的增益 g 及矩阵 sos，得出零点 z、极点 p 和增益 k。

【例 3-5】sos2zp 用法。

```
>> sos = [1   1   1   1   0 -1; -2   3   1   1 10   1];
[z,p,k] = sos2zp(sos)
```

运行程序，输出如下：

```
z =
   -0.5000 + 0.8660i
   -0.5000 - 0.8660i
    1.7808
   -0.2808
p =
   -1.0000
    1.0000
   -9.8990
   -0.1010
k =
    -2
```

（7）ss2sos 函数。功能：将状态方程形式转变为二阶分割形式。其调用格式如下：

[sos,g] = ss2sos(A,B,C,D)：由单输入单输出系统的状态方程矩阵 a，b，c 和 d 得到二阶分割形式的增益 g 及矩阵 sos。系统的零极点必须是共轭的，必须处于稳定状态。

[sos,g] = ss2sos(A,B,C,D,iu)：在由多输入单输出系统的状态形式向二阶分割形式转换中使用第 iu 个输入。

二阶分割形式参见 sos2ss。G 代表系统的全增益，如果 g 没有限定，则包含在第一阶当中。第二阶结构表示系统为 H(z)=G*H1(z)*H2(z)*…*HL(z)。

[sos,g] = ss2sos(A,B,C,D,'order')：限定二阶分割形式的排列，当 order 为'up'时，第一列包含最接近于原点的极点，最后一列包含最接近于单位圆的极点。当 order 为'down'时，阶数被按相反顺序排列。第一列包含最接近于原点的极点，最后一列包含最接近于单位圆的极点。零点与最接近于它们的极点配对。默认状态 order 为'up'。

[sos,g] = ss2sos(A,B,C,D,iu,'order')：指定行序列。

[sos,g] = ss2sos(A,B,C,D,iu,'order','scale')：限定预定的增益的缩放比例与二阶分割形式的分子系数。Scale 可以为'none'，'inf'或'two'，分别代表无、无限及两倍，默认为'none'。当 scale 为'inf'，order 为'up'，排列将最小化溢出的概率，另外当 scale 为'two'，order 为'down'，排列将最小化峰值的舍入噪声。

【例 3-6】ss2sos 用法。

```
>> [A,B,C,D] = butter(5,0.2);
sos = ss2sos(A,B,C,D)
```

运行程序，输出如下：

```
sos =
    0.0013    0.0013         0    1.0000   -0.5095         0
    1.0000    2.0017    1.0017    1.0000   -1.0966    0.3554
    1.0000    1.9955    0.9955    1.0000   -1.3693    0.6926
```

（8）ss2tf 函数。功能：将状态方程形式转变为传递函数形式。其调用格式如下：

[b,a] = ss2tf(A,B,C,D,iu)：将状态方程形式转换为传递函数形式，iu 用于指定变换所使用的

输入数。

（9）ss2zp 函数。功能：将状态方程形式转变为零极点增益形式。其调用格式如下：

[z,p,k] = ss2zp(A,B,C,D,i)：将状态方程形式(A,B,C,D)转变为零极点增益形式[z,p,k]，i 用于指定变换所使用的输入数。

【例 3-7】已知状态方程

$$H(z)=\frac{2+3z^{-1}}{1+0.4z^{-1}+z^{-2}}$$

求其传递函数形式。

其实现的 MATLAB 程序代码如下：

```
>> b = [2 3 0];
a = [1 0.4 1];
[z,p,k] = tf2zp(b,a)
z =
          0
    -1.5000
p =
   -0.2000 + 0.9798i
   -0.2000 - 0.9798i
k =
     2
>> [A,B,C,D] = tf2ss(b,a);
[z,p,k] = ss2zp(A,B,C,D,1)
z =
    -1.5000
          0
p =
   -0.2000 + 0.9798i
   -0.2000 - 0.9798i
k =
     2
```

（10）tf2ss 函数。功能：将传递函数形式转变为状态方程形式。其调用格式如下：

[A,B,C,D] = tf2ss(b,a)：将单信号输入系统传递函数形式（num，den）转变为状态方程形式[a, b, c, d]。分母系数向量中的分母系数是按 s 的降幂排列。系统传递函数分子系数矩阵 num 与系统输出具有一样的列数。[a, b, c, d]是按照控制语言的形式返回，计算同样适用于离散系统。在离散系统中为避免冲突，在使用该函数时对分子多项式进行补零以使其分母具有同等长度。

【例 3-8】已知某传递函数

$$H(s)=\frac{\begin{bmatrix}2s+3\\ s^2+2s+1\end{bmatrix}}{s^2+0.4s+1}$$

求其状态方程形式。

其实现的 MATLAB 程序代码如下：

```
>> b = [0 2 3; 1 2 1];
a = [1 0.4 1];
[A,B,C,D] = tf2ss(b,a)
```

运行程序，输出如下：

```
A =
   -0.4000   -1.0000
    1.0000         0
B =
     1
     0
C =
    2.0000    3.0000
    1.6000         0
D =
     0
     1
```

（11）tf2zp 函数。功能：将传递函数形式转变为零极点增益形式。其调用格式如下：

[z,p,k] = tf2zp(b,a)：将系统传递函数形式（b, a）转变为零极点增益形式[z,p,k]。分母系数向量中的分母系数是按 s 的降幂排列。系统传递函数分子系数矩阵 num 与系统输出具有一样的列数。[a, b, c, d]是按照控制语言的形式返回，计算同样适用于离散系统。在离散系统中为避免冲突，在使用该函数时对分子多项式进行补零以使其与分母具有同等长度。零点被返回于行向量 z，极点被返回于行向量 p，增益返回于向量 k。

【例 3-9】设计一个切比雪夫 I 型高通滤波器，使其通带截止频率为 80Hz，阻带截止频率为 60Hz，rp=1，rs=40，采样频率为 200Hz，分别表示为传递函数形式、格式滤波器形式、二阶分割形式、零极点增益形式以及状态方程形式。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=80;ws=60;
rp=1;rs=40;Fs=200;
[N,wn]=cheb1ord(wp/(Fs/2),ws/(Fs/2),rp,rs,'z');
[num,den]=cheby1(N,rp,wn,'high')
[k,v]=tf2latc(num,den)
sos=tf2sos(num,den)
[z,p,k]=tf2zp(num,den)
[A,B,C,D]=tf2ss(num,den)
```

运行程序，输出如下：

```
num =
    0.0003   -0.0015    0.0029   -0.0029    0.0015   -0.0003
den =
    1.0000    3.9634    6.6990    5.9815    2.8111    0.5558
k =
    0.9161
    0.9411
    0.9313
    0.8801
    0.5558
v =
    0.0048
```

```
    -0.0161
     0.0186
    -0.0101
     0.0026
    -0.0003
sos =
    0.0003   -0.0003         0    1.0000    0.8280         0
    1.0000   -2.0018    1.0018    1.0000    1.5944    0.7458
    1.0000   -1.9993    0.9993    1.0000    1.5410    0.9000
z =
    1.0009 + 0.0006i
    1.0009 - 0.0006i
    0.9997 + 0.0010i
    0.9997 - 0.0010i
    0.9989
p =
   -0.7705 + 0.5535i
   -0.7705 - 0.5535i
   -0.8280
   -0.7972 + 0.3321i
   -0.7972 - 0.3321i
k =
   2.9206e-004
A =
    -3.9634   -6.6990   -5.9815   -2.8111   -0.5558
     1.0000         0         0         0         0
          0    1.0000         0         0         0
          0         0    1.0000         0         0
          0         0         0    1.0000         0
B =
     1
     0
     0
     0
     0
C =
    -0.0026    0.0010   -0.0047    0.0006   -0.0005
D =
   2.9206e-004
```

（12）zp2sos 函数。功能：将零极点增益形式转变为二阶分割形式。其调用格式如下：

[sos,g] = zp2sos(z,p,k,'order')：确定二阶分割形式的排列顺序，如果 order 等于'up'，则第一列将包含最接近于原点的极点，最后一列将包含最接近于单位圆的极点；如果 order 等于'down'，阶数按照相反的顺序进行排列。零点总是与最接近于它的极点相配合。Order 默认为'up'。

[sos,g] = zp2sos(z,p,k,'order', 'scale')：限定预定的的增益缩放比例，与二阶分割形式的分子系数。Scale 可以为'none'，'inf'或'two'，分别代表无、无限及两倍，默认为'none'。当 scale 为'inf'，

order 为'up'，排列将最小化溢出的概率，另外当 scale 为'two'，order 为'down'，排列将最小化峰值的舍入噪声。

（13）zp2ss 函数。功能：将零极点增益形式转变为状态方程形式。其调用格式如下：

[A,B,C,D] = zp2ss(z,p,k)：由零极点增益形式得到状态方程形式。

（14）zp2tf 函数。功能：将零极点增益形式转变为传递函数形式。其调用格式如下：

[b,a] = zp2tf(z,p,k)：由零极点增益形式得到传递函数形式。

【例 3-10】设计一个椭圆带阻滤波器，使其通带频率范围 650Hz～850Hz，阻带频率范围为 700Hz～800Hz，rp=0.1，rs=50，采样频率为 2000Hz，分别表示为传递函数形式、格式滤波器形式、二阶分割形式、零极点增益形式以及状态方程形式。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp1=350;wp2=850;
ws1=700;ws2=800;
rp=0.1;rs=50;Fs=2000;
wp=[wp1,wp2];ws=[ws1,ws2];
[N,wn]=ellipord(wp/(Fs/2),ws/(Fs/2),rp,rs,'z');
[z,p,k]=ellip(N,rp,rs,wn,'stop')                %阻带
[sos,g]=zp2sos(z,p,k)
[A,B,C,D]=zp2ss(z,p,k)
[num,den]=zp2tf(z,p,k)
```

运行程序，输出如下：

```
z =
   0.2610 + 0.9653i   0.2610 - 0.9653i   0.1303 + 0.9915i   0.1303 - 0.9915i
  -0.2142 + 0.9768i  -0.2142 - 0.9768i  -0.6166 + 0.7873i  -0.6166 - 0.7873i
  -0.8349 + 0.5503i  -0.8349 - 0.5503i  -0.7887 + 0.6147i  -0.7887 - 0.6147i
p =
   0.4098 + 0.8641i   0.4098 - 0.8641i   0.4178 + 0.7050i   0.4178 - 0.7050i
   0.4129 + 0.3179i   0.4129 - 0.3179i  -0.7414 + 0.1821i  -0.7414 - 0.1821i
  -0.8223 + 0.3819i  -0.8223 - 0.3819i  -0.8643 + 0.4560i  -0.8643 - 0.4560i
k =
    0.0596
sos =
    1.0000    0.4285    1.0000    1.0000   -0.8258    0.2716
    1.0000    1.2331    1.0000    1.0000    1.4828    0.5829
    1.0000   -0.2606    1.0000    1.0000   -0.8356    0.6716
    1.0000    1.5775    1.0000    1.0000    1.6447    0.8221
    1.0000   -0.5219    1.0000    1.0000   -0.8197    0.9146
    1.0000    1.6699    1.0000    1.0000    1.7285    0.9549
g =
    0.0596
A =
  Columns 1 through 7
   -1.7285   -0.9772         0         0         0         0         0
    0.9772         0         0         0         0         0         0
   -0.0587    0.0461   -1.6447   -0.9067         0         0         0
```

```
         0         0    0.9067         0         0         0         0
   -0.0587    0.0461   -0.0672    0.1962   -1.4828   -0.7634         0
         0         0         0         0    0.7634         0         0
   -0.0587    0.0461   -0.0672    0.1962   -0.2497    0.5464    0.8197
         0         0         0         0         0         0    0.9564
   -0.0587    0.0461   -0.0672    0.1962   -0.2497    0.5464    1.2482
         0         0         0         0         0         0         0
   -0.0587    0.0461   -0.0672    0.1962   -0.2497    0.5464    1.2482
         0         0         0         0         0         0         0
  Columns 8 through 12
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
   -0.9564         0         0         0         0
         0         0         0         0         0
    0.0893    0.8258   -0.5211         0         0
         0    0.5211         0         0         0
    0.0893    0.5652    1.3978    0.8356   -0.8195
         0         0         0    0.8195         0
B =     1     0     1     0     1     0     1     0     1     0     1     0
C =
  Columns 1 through 7
   -0.0035    0.0027   -0.0040    0.0117   -0.0149    0.0326    0.0744
  Columns 8 through 12
    0.0053    0.0337    0.0833    0.0187    0.0239
D =
    0.0596
num =
  Columns 1 through 7
    0.0596    0.2459    0.6466    1.2331    1.9185    2.4544    2.6742
  Columns 8 through 13
    2.4544    1.9185    1.2331    0.6466    0.2459    0.0596
den =
  Columns 1 through 7
    1.0000    2.3750    2.0660    1.8350    3.1549    2.6819    0.8152
  Columns 8 through 13
    0.7752    1.0360    0.2010   -0.1330    0.0896    0.0763
```

3.2　系统的状态变量分析

3.2.1　状态方程与系统函数之间的转换

连续时间系统的状态方程与输出方程用矩阵可表示为

$$\begin{cases}\dot{\boldsymbol{\lambda}}(t)=\boldsymbol{A}\boldsymbol{\lambda}(t)+\boldsymbol{B}\boldsymbol{x}(t)\\ \boldsymbol{y}(t)=\boldsymbol{C}\boldsymbol{\lambda}(t)+\boldsymbol{D}\boldsymbol{x}(t)\end{cases} \tag{3-2}$$

离散时间 LTI 系统的状态方程与输出方程用矩阵可表示为

$$\begin{cases}\boldsymbol{\lambda}(n+1)=\boldsymbol{A}\boldsymbol{\lambda}(n)+\boldsymbol{B}\boldsymbol{x}(n)\\ \boldsymbol{y}(n)=\boldsymbol{C}\boldsymbol{\lambda}(n)+\boldsymbol{D}\boldsymbol{x}(n)\end{cases} \tag{3-3}$$

MATLAB 控制系统工具箱提供了 ss2tf 和 tf2ss 两个函数，来实现系统的状态空间（ss）表示法和系统函数（tf）表示法之间的互换。Tf2ss 函数是将一个系统的系统函数转化为状态空间表示法。其用法前面已经介绍过。在此不再展开介绍。

【例 3-11】已知某连续系统的系统函数为

$$H(s)=\frac{4s+10}{s^3+8s^2+19s+12}$$

试用 MATLAB 命令求系统的状态方程与输出方程。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[A,B,C,D]=tf2ss([4,10],[1,8,19,12])
A =
    -8   -19   -12
     1     0     0
     0     1     0
B =
     1
     0
     0
C =
     0     4    10
D =
     0
```

所以，系统状态方程与输出方程分别如下：

$$\begin{bmatrix}\dot{\lambda}_1(t)\\ \dot{\lambda}_2(t)\\ \dot{\lambda}_3(t)\end{bmatrix}=\begin{bmatrix}-8 & -9 & -10\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\\ \lambda_3(t)\end{bmatrix}+\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}\boldsymbol{x}(t)$$

$$\boldsymbol{y}(t)=[0 \quad 4 \quad 10]\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\\ \lambda_3(t)\end{bmatrix}$$

ss2tf 函数是将一个系统的状态空间表示法转换为系统函数，其用法前面已经介绍过。

【例 3-12】已知某离散时间系统的状态方程和输出方程分别为

$$\begin{bmatrix}\lambda_1(n+1)\\ \lambda_2(n+1)\end{bmatrix}=\begin{bmatrix}0 & 1\\ -3 & -4\end{bmatrix}\begin{bmatrix}\lambda_1(n)\\ \lambda_2(n)\end{bmatrix}+\begin{bmatrix}0\\ 2\end{bmatrix}x(n)$$

$$y(n)=[-1 \quad -2]\begin{bmatrix}\lambda_1(n)\\ \lambda_2(n)\end{bmatrix}+x(n)$$

试用 MATLAB 命令求该离散时间系统的系统函数 H（z）。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
A=[0 1;-3 -4];
B=[0;2];
C=[-1 2];
D=1;
[num,den]=ss2tf(A,B,C,D)
num =
      1.0000       8.0000       1.0000
den =
       1       4       3
>> Hz=tf(num,den,-1)     %写出离散时间系统的函数 H(z)
  Transfer function:
z^2 + 8 z + 1
-------------
z^2 + 4 z + 3
  Sampling time: unspecified
```

即系统函数为

$$H(z)=\frac{1+8z^{-1}+z^{-2}}{1+4z^{-1}+3z^{-2}}$$

【例 3-13】一个多输入多输出系统，其状态方程和输出方程分别为

$$\begin{bmatrix}\dot{\lambda}_1(t)\\ \dot{\lambda}_2(t)\end{bmatrix}=\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}+\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

$$\begin{bmatrix}y_1(t)\\ y_2(t)\\ y_3(t)\end{bmatrix}=\begin{bmatrix}1 & 0\\ 1 & 1\\ 0 & 2\end{bmatrix}\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}+\begin{bmatrix}0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

试用 MATLAB 命令求该系统的系统函数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
A=[0 1;-2 -3];
B=[1 0;1 1];
C=[1 0;1 1;0 2];
D=[0 0;1 0;0 1];
[num1,den1]=ss2tf(A,B,C,D,1)
num1 =
            0       1.0000       4.0000
       1.0000       5.0000       4.0000
            0       2.0000      -4.0000
den1 =
       1       3       2
>> [num2,den2]=ss2tf(A,B,C,D,2)
num2 =
            0       0.0000       1.0000
```

```
            0    1.0000    1.0000
       1.0000    5.0000    2.0000
den2 =
     1     3     2
```

所以，系统函数为

$$H(s)=\frac{1}{s^2+3s+2}\begin{bmatrix} s+4 & 1 \\ s^2+5s+4 & s+1 \\ 2s-4 & s^2+5s+2 \end{bmatrix}$$

3.2.2 状态方程的变换域符号求解分析

对连续系统而言，状态方程可通过拉普拉斯变换法求解。状态方程是一阶微分方程组，求解状态方程时必须知道状态 t=0_时刻的状态值。系统状态方程和输出方程可分别表示为

$$\begin{cases} \dot{\lambda}(t)=A\lambda(t)+Bx(t) \\ y(t)=C\lambda(t)+Dx(t) \\ \lambda(0_-) \end{cases} \tag{3-4}$$

对式（3-4）进行拉普拉斯变换（下面章节将展开介绍），整理得

$$\boldsymbol{\Lambda}(s)=(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{\lambda}(0_-)+(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{X}(s) \tag{3-5}$$

式中：$\boldsymbol{I}$ 为单位矩阵；$\boldsymbol{\Lambda}(s)$、$\boldsymbol{X}(s)$分别为状态向量$\boldsymbol{\lambda}(t)$和激励信号向量 $\boldsymbol{x}(s)$通过拉普拉斯变换所得到的。式（3-5）即为状态方程的拉普拉斯变换解。

将式（3-5）代入经过拉普拉斯变换后的输出方程，得

$$Y(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{\lambda}(0_-)+[\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})\boldsymbol{B}+\boldsymbol{D}]\boldsymbol{X}(s) \tag{3-6}$$

式中：$Y(s)$为输出信号向量 $\boldsymbol{y}(t)$的拉普拉斯变换。式（3-6）中的第一项对应系统零输入响应的拉普拉斯变换；第二项对应系统零状态响应的拉普拉斯变换。

定义矩阵

$$\boldsymbol{\Phi}(s)=(s\boldsymbol{I}-\boldsymbol{A})^{-1} \tag{3-7}$$

则系统函数矩阵为

$$\boldsymbol{H}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}=\boldsymbol{C}\boldsymbol{\Phi}(s)\boldsymbol{B}+\boldsymbol{D} \tag{3-8}$$

利用 MATLAB 强大的矩阵运算功能和符号运算功能，可以方便地求解系统方程。

【例 3-14】已知连续系统的状态方程和输出方程分别为

$$\begin{bmatrix} \dot{\lambda}_1(t) \\ \dot{\lambda}_2(t) \end{bmatrix}=\begin{bmatrix} -1 & -4 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \lambda_1(t) \\ \lambda_2(t) \end{bmatrix}+\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}=\begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \lambda_1(t) \\ \lambda_2(t) \end{bmatrix}+\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

其初始状态和激励信号分别如下：

$$\begin{bmatrix}\lambda_1(0_-)\\ \lambda_2(0_-)\end{bmatrix}=\begin{bmatrix}2\\1\end{bmatrix}$$

$$\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}=\begin{bmatrix}u(t)\\ \mathrm{e}^{-t}u(t)\end{bmatrix}$$

试用 MATLAB 命令求系统的状态变量和输出响应。

```
>> clear all;
syms s
A=[-1 -4;1 -1];
B=[0 1;1 0];
C=[1 1;0 -1];
D=[1 0;1 0];
r0=[2;1];
X=[1/s;1/(s+1)];                          %激励信号的拉普拉斯变换
phis=inv(s*eye(2)-A);                     %求状态变量的拉普拉斯变换
rs=phis*(r0+B*X);
rs=simplify(rs)
rs =
                  (2*s^2-s-4)/(s^2+2*s+5)/s
 (5*s^2+6*s+s^3+1)/(s^2+2*s+5)/(s+1)/s
>>rt=ilaplace(rs);                        %求状态变量时域解
rt=simplify(rt)
rt =
                -4/5+14/5*exp(-t)*cos(2*t)-11/10*exp(-t)*sin(2*t)
 1/5+11/20*exp(-t)*cos(2*t)+7/5*exp(-t)*sin(2*t)+1/4*exp(-t)
>>yt=C*phis*r0+[C*phis*B+D]*X;             %求输出响应的拉普拉斯变换
ys=simplify(yt)
ys =
   (4*s^3+9*s^2+8*s+2)/(s^2+2*s+5)/s/(s+1)
          -(2*s^2-s-4)/(s^2+2*s+5)/s/(s+1)
>> yt=ilaplace(ys);                       %求输出响应的时域解
yt=simplify(yt)
yt =
 2/5+1/4*exp(-t)+67/20*exp(-t)*cos(2*t)+3/10*exp(-t)*sin(2*t)
   4/5-11/20*exp(-t)*cos(2*t)-7/5*exp(-t)*sin(2*t)-1/4*exp(-t)
```

用 ilaplace 求拉普拉斯反变换时假设了信号是因果的，因此，状态变量的结果暗含着与单位阶跃信号相乘，即

$$\begin{bmatrix}\lambda_1(t)\\ \lambda_2(t)\end{bmatrix}=\begin{bmatrix}-\dfrac{4}{5}u(t)+\mathrm{e}^{-t}\left(\dfrac{14}{5}\cos(2t)-\dfrac{11}{10}\sin(2t)\right)u(t)\\ \dfrac{1}{5}u(t)+\mathrm{e}^{-t}\left(\dfrac{1}{4}+\dfrac{11}{20}\cos(2t)+\dfrac{7}{5}\sin(2t)\right)u(t)\end{bmatrix}$$

输出响应如下：

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{2}{5}u(t) + \mathrm{e}^{-t}\left(\frac{1}{4} + \frac{67}{20}\cos(2t) + \frac{3}{10}\sin(2t)\right)u(t) \\ \frac{4}{5}u(t) - \mathrm{e}^{-t}\left(\frac{1}{4} + \frac{11}{20}\cos(2t) + \frac{7}{5}\sin(2t)\right)u(t) \end{bmatrix}$$

根据所得结果可画出输出响应波形，其 MATLAB 程序代码如下：

```
>> t=0:0.01:4;
y1=2/5+1/4*exp(-t)+67/20*exp(-t).*cos(2*t)+3/10*exp(-t).*sin(2*t);
y2=4/5-1/4*exp(-t)-11/20*exp(-t).*cos(2*t)-7/5*exp(-t).*sin(2*t);
subplot(2,1,1);plot(t,y1);
grid on;
ylabel('y1(t)');xlabel('t');
subplot(2,1,2);plot(t,y2);
grid on;
ylabel('y2(t)');xlabel('t');
```

运行程序，效果如图 3-1 所示。

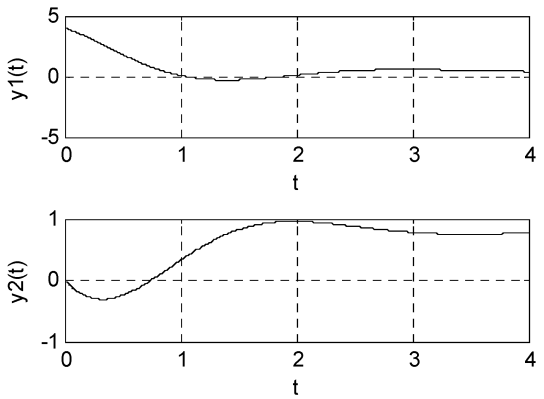


图 3-1　连续系统的输出响应

与连续系统类似，离散系统的状态方程和输出方程可描述为

$$\begin{cases} \lambda(n+1) = A\lambda(n) + Bx(t) \\ y(n) = C\lambda(n) + Dx(n) \\ \lambda(0) \end{cases} \tag{3-9}$$

离散时间系统状态方程可通过 Z 变换法求解。对式（3-9）进行 Z 变换，整理得

$$\Lambda(z)=(z\boldsymbol{I}-\boldsymbol{A})^{-1}z\ \boldsymbol{\lambda}(0)+(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{X}(z) \tag{3-10}$$

式中：$\boldsymbol{I}$ 为单位矩阵；$\Lambda(z)$、$X(z)$分别为状态向量$\boldsymbol{\lambda}(n)$和激励信号向量 $\boldsymbol{x}(n)$的 Z 变换。式（3-10）即为离散时间系统状态方程的 Z 变换解。

将式（3-10）代入经过 Z 变换后的输出方程，得

$$Y(z)=\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}z\ \boldsymbol{\lambda}(0)+[\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}]X(z) \tag{3-11}$$

式中：$Y(z)$为输出信号向量 $\boldsymbol{y}(n)$的 Z 变换。

式（3-11）中的第一项对应系统零输入响应的 Z 变换；第二项对应系统零状态响应的 Z 变换。所以，系统函数矩阵为

$$\boldsymbol{H}(z)=\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D} \tag{3-12}$$

【例 3-15】给定系统状态方程、输出方程、激励信号和系统的初始条件分别为

$$\begin{bmatrix}\lambda_1(n+1)\\ \lambda_2(n+1)\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\frac{1}{6} & \frac{5}{6}\end{bmatrix}\begin{bmatrix}\lambda_1(n)\\ \lambda_2(n)\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}x(n)$$

$$y(n)=[-1 \quad 5]\begin{bmatrix}\lambda_1(n)\\ \lambda_2(n)\end{bmatrix}$$

$$x(n)=u(n),\quad \begin{bmatrix}\lambda_1(0)\\ \lambda_2(0)\end{bmatrix}=\begin{bmatrix}2\\ 3\end{bmatrix}$$

试用 MATLAB 命令求输出响应 $y(n)$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
syms z
A=[0 1;-1/6,5/6];B=[0;1];
C=[-1 5];D=0;
r0=[2;3];
X=z/(z-1);                          %激励信号的 Z 变换
phiz=inv(z*eye(2)-A);
yz=C*phiz*z*r0+(C*phiz*B+D)*X;      %求输出响应的 Z 变换
yz=simplify(yz)
yz =
6*z*(13*z^2-11*z+2)/(6*z^2-5*z+1)/(z-1)
>> yn=iztrans(yz)                   %求输出响应的时域解
yn =
        -2*(1/3)^n+3*(1/2)^n+12
```

同样，用 iztrans 函数求 Z 反变换时假设了信号是因果的，因此，求得的结果暗含着与单位阶跃序列相乘，即输出响应为

$$y(n)=\left[3\times\left(\frac{1}{2}\right)^n-2\times\left(\frac{1}{3}\right)^n+12\right]u(n)$$

根据所得结果可画出输出响应序列的波形，其实现的 MATLAB 程序代码如下：

```
>> n=0:15;
yn=3*(1/2).^n-2*(1/3).^n+12;
stem(n,yn);
grid on;
xlabel('n');ylabel('y(n)');
axis([0 15 11 14]);
```

根据程序运行结果，绘出 $0\leqslant n\leqslant 15$ 范围内该离散时间系统输出响应序列的波形，如图 3-2 所示。

另外，通过式（3-8）与式（3-11）可方便地由系统方程的矩阵求得系统函数矩阵。例 3-13 的 MATLAB 源程序如下：

```
>> clear all;
syms s
```

```
A=[0 1;-2 -3];
B=[1 0;1 1];
C=[1 0;1 1;0 2];
D=[0 0;1 0;0 1];
phis=inv(s*eye(2)-A);
Hs=(C*phis*B+D);
Hs=simplify(Hs)
Hs =
[          (s+4)/(s^2+3*s+2),                1/(s^2+3*s+2)]
[                  (s+4)/(s+2),                      1/(s+2)]
[     2*(-2+s)/(s^2+3*s+2), (5*s+s^2+2)/(s^2+3*s+2)]
```

比较例 3-13，可知计算结果一样。

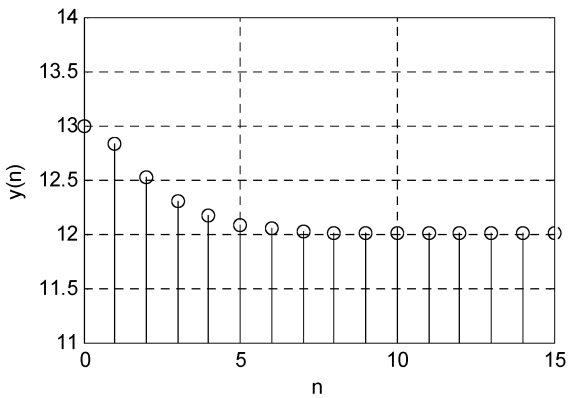


图 3-2　离散系统的输出响应

3.2.3　状态方程的时域符号求解分析

时域法求解状态方程过程要用到矩阵指数函数 e^{At}，对式（3-4）所描述的连续时间系统，通过推导可得系统状态方程的解为

$$\boldsymbol{\lambda}(t)=e^{\boldsymbol{A}t}\boldsymbol{\lambda}(0\_)+e^{\boldsymbol{A}t}\boldsymbol{B}*\boldsymbol{x}(t) \tag{3-13}$$

式（3-13）的解包含两部分：第一部分 $e^{\boldsymbol{A}t}\boldsymbol{\lambda}(0_)$是系统状态变量的零输入解；第二部分 $e^{\boldsymbol{A}t}\boldsymbol{B}*\boldsymbol{x}(t)$是系统状态变量的零起始状态解。

通过进一步推导还可得到系统的完全响应为

$$\boldsymbol{y}(t)=\boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{\lambda}(0\_)+[\boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{B}+\boldsymbol{D}\boldsymbol{\delta}(t)]*x(t) \tag{3-14}$$

式中：第一项为系统的零输入响应；第二项为的零状态响应。

从式（3-14）可知系统的冲激响应为

$$\boldsymbol{h}(t)=\boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{B}+\boldsymbol{D}\boldsymbol{\delta}(t) \tag{3-15}$$

根据以上结论，利用 MATLAB 可以完成时域求解状态方程和输出方程。MATLAB 符号工具箱提供了 expm 函数，可利用它来求矩阵指数函数 $e^{\boldsymbol{A}t}$。

【例 3-16】试用 MATLAB 时域求解法求例 3-14 中连续时间系统的状态变量、冲激响应和输出响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
syms t
```

```
A=[-1 -4;1 -1];
B=[0 1;1 0];
C=[1 1;0 -1];
D=[1 0;1 0];
x=[Heaviside(t);exp(-t)*Heaviside(t)];
r0=[2;1];
E=expm(t*A)                          %求解矩阵指数函数
E =
[      exp(-t)*cos(2*t),    -2*exp(-t)*sin(2*t)]
[ 1/2*exp(-t)*sin(2*t),       exp(-t)*cos(2*t)]
>> rzi=E*r0;                         %状态方程零输入解
%状态方程零状态解,利用 s 域乘法的拉普拉斯反变换求时域卷积
>> rzs=ilaplace(laplace(E*B)*laplace(x));
>> rt=simplify(rzi+rzs)              %状态变量完全解
rt =
                14/5*exp(-t)*cos(2*t)-11/10*exp(-t)*sin(2*t)-4/5
  7/5*exp(-t)*sin(2*t)+11/20*exp(-t)*cos(2*t)+1/5+1/4*exp(-t)
>> ht=C*E*B+D*Dirac(t)               %求冲激响应
ht =
[-2*exp(-t)*sin(2*t)+exp(-t)*cos(2*t)+dirac(t), exp(-t)*cos(2*t)+1/2*exp(-t)*sin(2*t)]
[-exp(-t)*cos(2*t)+dirac(t)                   ,                -1/2*exp(-t)*sin(2*t)]
>> yzi=C*B*r0;                       %输出零输入解
%输出零状态解,利用 s 域乘法的拉普拉斯反变换求时域卷积
>> yzs=ilaplace(laplace(ht)*laplace(x));
>> yt=simplify(yzi+yzs)              %输出完全解
yt =
  17/5+7/20*exp(-t)*cos(2*t)+13/10*exp(-t)*sin(2*t)+1/4*exp(-t)
     -6/5+9/20*exp(-t)*cos(2*t)-2/5*exp(-t)*sin(2*t)-1/4*exp(-t)
```

比较例 3-14 和例 3-16，不难发现计算结果是相同的。

对于式（3-9）所描述的离散时间系统，也可以推导其状态方程的解，即为

$$\boldsymbol{\lambda}(n)=\boldsymbol{A}^{n}\boldsymbol{\lambda}(0)+\boldsymbol{A}^{n-1}\boldsymbol{u}(n-1)*\boldsymbol{B}\boldsymbol{x}(n) \tag{3-16}$$

则输出响应序列为

$$\boldsymbol{y}(n)=\boldsymbol{C}\boldsymbol{A}^{n}\boldsymbol{\lambda}(0)+[\boldsymbol{C}\boldsymbol{A}^{n-1}\boldsymbol{B}\boldsymbol{u}(n-1)+\boldsymbol{D}\boldsymbol{\delta}(n)]*\boldsymbol{x}(n) \tag{3-17}$$

式中：第一项为系统的零输入响应；第二项为系统的零状态响应。

系统的单位取样响应为

$$\boldsymbol{h}(n)=\boldsymbol{C}\boldsymbol{A}^{n-1}\boldsymbol{B}\boldsymbol{u}(n-1)+\boldsymbol{D}\boldsymbol{\delta}(n) \tag{3-18}$$

$\boldsymbol{A}^{n}$ 计算要利用式 $\boldsymbol{A}^{n}=\boldsymbol{Z}^{-1}[(\boldsymbol{I}-\boldsymbol{z}^{-1}\boldsymbol{A})^{-1}]$。由于 MATLAB 符号工具箱没有提供符号卷积函数，在求状态变量和输出响应序列的零状态解时也要采用 Z 变换法来求解。与离散时间系统状态方程的 Z 变换法相比较，不难发现，在 MATLAB 中离散时间系统状态方程的时域符号求解与变换域求解是统一的。

3.2.4　系统方程的数值求解分析

当一个系统用状态空间表示法来表示时，可利用 MATLAB 的 lsim 函数求解系统的响应。连续时间系统在前面已经介绍过，在此不再展开介绍。

【例 3-17】试用 MATLAB 数值求解法求例 3-14 中连续时间系统的输出响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0:0.01:4;
A=[-1 -4;1 -1];
B=[0 1;1 0];
C=[1 1;0 -1];
D=[1 0;1 0];
r0=[2;1];
f(:,1)=ones(length(t),1);          %第一个输入 u(t)在 t 上的样值的列向量
f(:,2)=exp(-t)';                   %第二个输入 exp-t)在 t 上的样值的列向量
sys=ss(A,B,C,D);                   %获取连续系统模型
y=lsim(sys,f,t,r0);                %数值求解系统模型
subplot(2,1,1);plot(t,y(:,1));
grid on;
xlabel('t');ylabel('y1(t)');
subplot(2,1,2);plot(t,y(:,2));
grid on;
xlabel('t');ylabel('y2(t)');
```

运行程序，效果如图 3-3 所示，将其与图 3-1 比较不难发现所得结果是一样的。

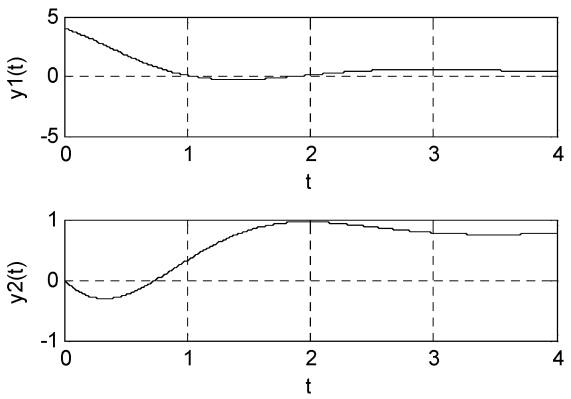


图 3-3　系统的输出响应数值求解

对离散时间系统而言，输入 u 的采样率应与系统本身的采样率相同，因此参数 t 就是冗余的，可以设为[]，即 empty 矩阵，因此，lsim 函数的语句格式如下：

y=lsim(sys, u, [], x0)

其中：sys 是由 sys=ss(A,B,C,D,[])获得的状态空间表示法所表示的离散时间系统模型。

【例 3-18】试用 MATLAB 数值求解法求例 3-15 中连续时间系统的输出响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=15;
n=0:1:N;
A=[0 1;-1/6,5/6];B=[0;1];
C=[-1 5];D=0;
r0=[2;3];
u=ones(1,N+1);              %输入序列 u(n)
```

```
sys=ss(A,B,C,D,[]);      %获取离散时间系统模型
yn=lsim(sys,u,[],r0);
stem(n,yn,'filled');
grid on;
xlabel('t');ylabel('y(n)');
axis([0 15 11 14]);
```

运行程序，结果如图 3-4 所示，将其与图 3-2 比较不难发现所得结果是一样的。

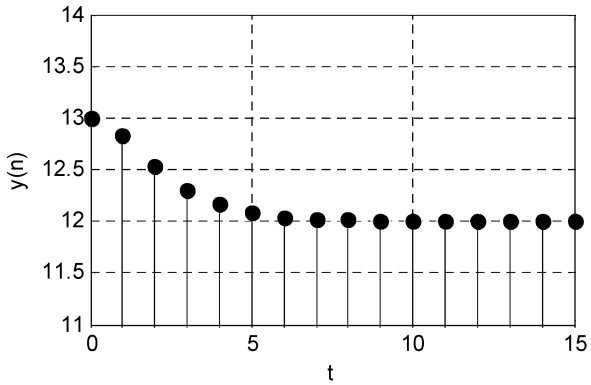


图 3-4　离散时间系统的输出响应

对于离散时间系统，还可直接利用式（3-9）递归求解系统方程的数值。例如，可由以下的 MATLAB 程序实现例 3-18，注意程序中的输入序列为 u(n)，在 n≥0 时取值为 1。其实现的 MATLAB 程序代码如下：

```
>> A=[0 1;-1/6,5/6];B=[0;1];
C=[-1 5];D=0;
r0=[2;3];
for n=0:15
    yn=C*r0;
    rnplus1=A*r0+B*1;   %状态变量递增 1
    r0=rnplus1;
    stem(n,yn,'filled');
    hold on;
end
grid on;
xlabel('t');ylabel('y(n)');
axis([0 15 11 14]);
hold off;
```

运行程序，效果同图 3-4 一样。

3.3　数据采集过程

3.3.1　创建一个设备对象

设备对象是用于访问硬件设备的工具箱组件。设备对象提供了硬件功能的控制通路，通过它用户可以控制数据采集应用系统的行为。每个设备对象都对应一个特定的硬件子系统。

设备对象需要通过调用对象创建函数来创建，对象创建函数是采用 MATLAB 提供的面向对象编程功能来创建的。数据采集工具箱中几个对象创建函数的名称及其功能如下：

① analoginput：创建一个模拟量输入对象。

② analogoutput：创建一个模拟量输出对象。

③ digitalio：创建一个数字量 I/O 对象。

在创建对象之前，应对相关的硬件驱动适配器进行注册，一般适配器注册是自动进行的。如果由于某种原因适配器没有自动注册，可以使用 daqregister 函数来手动注册。

用户可以通过 daqhwinfo 函数的 ObjectConstructorName 属性来查看如何为一特定的硬件子系统创建设备对象。例如，为查看如何为 NI 硬件板卡创建一个模拟量输入对象，应将板卡的适配器名称作为 daqhwinfo 函数的输入参数。

1．创建一个设备对象数组

在 MATLAB 中，用户可以将单个的变量整合在一起形成数组，对于设备对象此法也是适用的。

假设分别为一声卡创建了模拟量输入对象 ui，模拟量输出对象 uo：

```
ui=analoginput('winsound');
uo=analogoutput('winsound');
```

可以根据 MATLAB 语法来创建由 ui 和 uo 元素组成的行向量 x：

```
x=[ui uo]
```

输出如下：

```
Index:    Subsystem:          Name:
1          Analog Input        winsound0-UI
2          Analog Output       winsound0-UO
```

也可以创建列向量 y：

```
y=[ui;uo]
```

输出同 x=[ui uo]一样的结果。

值得注意的是，在 MATLAB 中用户不能创建设备对象矩阵。

如输入：

```
>> z=[ui uo;ui uo]
```

输出为：

```
??? Error using ==> analoginput.vertcat at 37
Only a row or column vector of device objects can be created.
```

设备对象只能创建为行向量或列向量。

在应用程序中，可以将设备对象向量作为函数的参数。

仅通过调用一个 set 函数来同时配置 ui 和 uo 对象的相同属性的属性值。例如：

```
set(x,'SampleRate',44100)
```

2．设备对象的存在位置

创建好的设备对象，在 MATLAB 工作空间和数据采集引擎中同时存在。

假设用户创建一个声卡的模拟量输入设备对象 ui，并定义了 ui 的一个复制 xiui：

```
ui=analoginput('winsound');
xiui=ui;
```

设备对象复制 xiui 与原始对象 ui 是一样的。也就是说，如果对 ui 的某个属性设定相应的值，那么 xiui 对象的相应属性也被赋予相应的属性值。例如：

```
set(ui,'SampleRate',16920)
get(xiui,'SampleRate')
ans =
        16920
```

由上面的例子可见，ui 和 xiui 返回相同的属性值是因为它们在数据采集引擎中映射的是同一个设备对象，如图 3-5 所示。

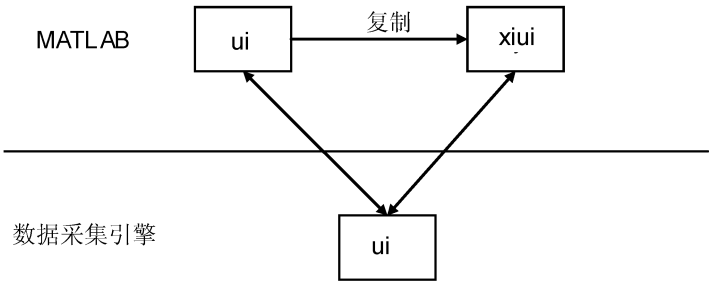


图 3-5　数据采集引擎映射

当删除原始设备对象和复制时，设备对象的引擎将被删除。这时对象在工作空间中的任何复制将不再可用，因为它们不再映射到任何硬件。这样的设备对象称作“无效对象”。例如在 MATLAB 中输入：

```
delete(ui);
xiui
```

将显示如下信息：

```
Invalid Data Acquisition object.
This object is not associated with any hardware and
should be removed from your workspace using CLEAR.
```

对于无效对象，用户应该使用清除命令将它们从工作空间中清除。

3.3.2　获取或输出数据

配置好设备对象之后，就可以获取或输出数据了。获取或输出数据包括如下步骤：

① 启动设备对象。

② 记录或发送数据。

③ 停止设备对象。

由于数据是在 MATLAB 和硬件之间传输的，所以可以把设备对象看作工作于某种特定的状态。在数据采集工具箱中定义了两种工作状态。

（1）运行状态：对于模拟量输入对象，运行状态是指从模拟量输入子系统中获取数据的状态，但是获取的数据并不一定要存入内存和磁盘文件中。对于模拟量输出对象，运行状态是指数据引擎中的数据队列已准备好输出到模拟量输出子系统中。

对于模拟量输入、输出对象，运行状态均由 Running 属性来标识，Running 属性的取值为 On 或 Off。

（2）记录或发送状态：对于模拟量输入对象，记录状态是指将从模拟量输入子系统获取的数据存储到引擎或磁盘文件中，记录状态由 Logging 属性来标识，Logging 属性的取值为 On 或 Off; 对于模拟量输出对象，发送状态是指数据引擎中的数据队列被输出到模拟量输出子系统中，发送状态由 Sending 属性标识，Sending 属性的取值为 On 或 Off。

Running，Logging 和 Sending 属性都是只读属性，由数据引擎自动赋值 On 或 Off。当 Running 属性为 Off 时，Logging 和 Sending 属性必是 Off，当 Running 属性为 On 时，Logging 和 Sending 属性只在触发时设置为 On。

注意：数字量 I/O 对象也具有运行状态。但是，由于数字量 I/O 对象不用在数据引擎中存储数据，所以它们记录和发送状态。

1．启动设备对象

用户可以使用 start 函数来启动一个设备对象。

【例 3-19】启动模拟量输入对象 ui。

```
ui=analoginput('winsound');
addchannel(ui,1:2);
start(ui)
```

当执行 start 函数后，Running 属性的属性值被自动设置为 On，并且设备对象和硬件设备在此时就会根据所配置的或默认的属性值运行。

在使用模拟量输入对象进行数据采集的过程中，可以使用 peckdata 函数来预览数据。peckdata 函数返回的仅是最新数据的映射，而不会将数据从数据引擎中移除。

【例 3-20】要预览模拟量输入对象 ui 中每个通道所获取的最近 500 个采样样本，代码如下：

```
data=peekdata(ui,500);
```

因为预览数据通常是一个低优先级的任务，所以，peekdata 函数不能确保返回所有期望的数据。在设备对象运行的任何时刻均可以预览数据。

2．记录或发送数据

在设备对象运行的过程中，可以：

① 将从模拟量输入子系统中获取的数据记录到数据引擎（内存）或磁盘文件中。

② 将数据引擎中的数据队列输出到模拟量输出子系统中。

但是，在记录或发送数据之前，必须有触发发生。配置模拟量输入、输出触发要用到 TriggerType 属性。在本章中的所有示例中 TriggerType 属性的属性值都采用的是默认值 Immediate，表示在 start 函数开始运行时触发立即执行。

3．提取记录数据

当模拟量输入对象的触发发生时，Logging 属性自动被设置为 On，同时由硬件获取的数据被记录到数据引擎或磁盘文件中。可以使用 getdata 函数来提取记录到引擎中的数据。从 ui 的每个通道提取 500 个样本的代码如下：

```
data=getdata(ui,500);
```

getdata 函数将锁定 MATLAB 命令行，直至所有的需要数据返回到 MATLAB 工作空间为止。在触发执行后的任何时刻均可以提取数据。

4．发送数据队列

对于模拟量输出对象，在数据输出到硬件之前，必须使用 putdata 函数在数据引擎中对数据进行排列。

【例 3-21】在数据中为模拟量输出对象 uo 的每个通道排列 8000 个样本：

```
uo=analogoutput('winsound');
addchannel(uo,1:2);
data=cos(linspace(0,2*pi*500,8000))';
putdata(uo,[data data])
```

```
start(uo)    %在数据队列可以输出之前,应运行模拟量输出对象
```

当触发发生时，Sending 属性自动被设置为 On，同时数据队列被发送到硬件中。

5．停止设备对象

在下列条件中的任何一个条件下均可停止模拟量输入与输出对象：

① 执行 stop 函数。

② 所需数量的样本已被获取或发送。

③ 发生了硬件 run-time 错误。

④ 超时错误。

当设备对象停止时，属性 Running、Logging 和 Sending 自动被设置为 Off。这时，可以对设备对象进行重新配置，或者使用当前配置来再次运行设备对象。

6．清除

当不再需要设备对象时，应该将对象从数据引擎（内存）以及 MATLAB 工作空间中清除。这时结束数据采集过程的最后步骤。

可以使用 delete 函数来将设备对象从内存中清除。例如，要清除前面几节中创建的模拟量输入对象 ui：

```
delete(ui)
```

被从数据引擎（内存）删除的设备对象将不再可用（即不再与硬件连接），所以应该使用 clear 命令将对象从 MATLAB 工作空间中清除。

```
clear ui
```

如果用户使用 clear 命令来删除一个与硬件保持连接的设备对象，该对象将从工作空间中移除但仍将保持与硬件的连接。可以使用 daqfind 函数将已删除的设备对象恢复到 MATLAB 工作空间。

【例 3-22】由 MATLAB 产生的余弦波数据通过声卡上的 D/A 转换器输出到扬声器。代码如下：

```
% 为声卡创建模拟量输出对象 uo
uo=analogoutput('winsound');
% 为 uo 添加一个通道
ch=addchannel(uo,1);
% 定义输出时间 4s，为基本安装属性分配值，排列产生的数据，并调用一次 putdata 函数将数据排列到对象 uo
duration=4;
set(uo,'SampleRate',8000);
set(uo,'TriggerType','Manual');
ActualRate=get(uo,'SampleRate');
len=ActualRate*duration;
data=cos(linspace(0,2*pi*500,len))';
putdata(uo,data);
% 执行 uo，触发触发器，等待设备对象停止运行
start(uo);
trigger(uo);
waittilstop(uo,5);
% 不再需要 uo 时，应将它从内存和 MATLAB 工作空间中清除
delete(uo);
```

```
clear uo;
```

3.4 函数参考

3.4.1 创建设备对象

在 MATLAB 中提供了许多函数实现创建设备对象，下面分别进行介绍。

1．analoginput 函数

功能：创建模拟量输入对象。其调用格式如下：

AI = analoginput('adaptor')：为 ID=0 的声卡创建模拟量输入对象 AI（'adaptor'必须为 winsound），这是唯一不需要 ID 的情况。

AI = analoginput('adaptor',ID)：为适配器名为'adptor'、设备标识符为 ID 的硬件设备创建模拟输入对象 AI。'adaptor'可以是 advantech、hpe1432、keithley、mcc、nidaq 和 winsound 等。ID 可以为整型数或字符串。

当创建好一个模拟量输入对象后，它不含有任何硬件通道。要运行设备对象，还必须使用 addchannel 函数来添加对象。

可以为同一个特定的模拟量输入子系统创建多个模拟量输入对象。但是，一次只能执行一个对象。

模拟量输入对象保存在数据采集引擎和 MATLAB 工作空间中。如果创建了一个设备对象的复制，那么复制将引用引擎中的原始设备对象。

如果 ID 为一个数值，那么可以将基设置为一个整型数和字符串，但是如果 ID 中含有非数字字符，则只能将 ID 设置字符串。

2．analogoutput 函数

功能：创建模拟量输出对象。

AO = analogoutput('adaptor')：为 ID=0 的声卡创建模拟量输出对象 AO（adaptor 必须为 winsound）。这是唯一不需要 ID 的情形。

AO = analogoutput('adaptor',ID)：为适配器名为 adaptor、设备标识符为 ID 的硬件设备创建模拟量输出对象 AO。ID 可以为整型数或字符串。

当创建好一个模拟量输出对象后，它不含有任何硬件通道。为运行设备对象，还必须使用 addchannel 函数来添加对象。

可以为同一个特定的模拟量输出子系统创建多个模拟量输出对象。但是，一次只能执行一个对象。

模拟量输出对象保存在数据采集引擎和 MATLAB 工作空间中。如果创建了一个设备对象的复制，那么复制将引用引擎中的原始设备对象。

如果 ID 为一个数值，那么可以将其设置为一个整型数和字符串。但是如果 ID 中含有非数字字符，则只能将 ID 设置字符串。

3．digitalio 函数

功能：创建数字量 I/O 对象。其调用格式如下：

DIO = digitalio('adaptor',ID)：为适配器为 adaptor、硬件设备的设备标识符为 ID 的硬件创建数字量 I/O 对象 DIO。ID 可以以整数和字符串形式指定。

3.4.2　获取并设置属性

在 MATLAB 中同时也提供了一些函数实现获取并设置函数参考的属性。下面分别进行介绍。

1．get 函数

返回设备对象的属性。其调用格式如下：

out=get(obj)：返回结构 out，结构中名称字段为 obj 的属性名，同时值字段为对应的属性的值。

out=get(obj.Channel(index))：返回结构 out，结构中名称字段为 obj 的通道的属性名，同时值字段为对应的属性的值。

out=get(obj.Line(index))：返回结构 out，结构中名称字段为 obj 的数据线的属性名，同时值字段为对应的属性的值。

out=get(obj,'PropertyName')：将由 PropertyName 指定的属性的值返回到 out 中。如果 PropertyName 为 1×n 或 n×1 的包含属性名的字符串数组，则将返回一个 1×n 的属性值数组到 out 中。如果 obj 为一数据获取对象数组。则 out 将是一个 m×n 的属性值数组，其中 m 等于 obj 的长度，n 等于所指定的属性数。

out=get(obj.Channel(index), 'PropertyName')：将 obj 所包含的指定通道的 PropertyName 指定的属性值返回到 out 中。如果指定了多个通道的多个属性，则 out 为一个 m×n 的数组，其中 m 为通道数，n 为属性数量。

out=get(obj.Line(index), 'PropertyName')：将 obj 所包含的指定数据线的 PropertyName 指定的属性的值返回到 out 中。如果指定了多条数据线的多个属性，则 out 为一个 m×n 的数组，其中 m 为数据线，n 为属性数量。

get(…)：显示指定的设置对象、通道或数据项的所有属性名和属性的当前值。首先显示的是基本属性，其次是设备特定的属性。

下列是一些用于返回属性值的方法：

```
chan=get(ui, 'Channel')
out=get(ui, {'SampleRate', 'TriggerDelayUnits'});
out=get(ui);
get(chan(1), 'Units');
get(chan, {'Index', 'HwChannel', 'ChannelName'});
```

【例 3-23】为对象创建多个属性。

```
>> patch;surface;text;line
output = get(get(gca,'Children'),props)
```

运行程序，输出如下，效果如图 3-6 所示。

```
output =
    'on'    'on'    'on'    'line'
    'on'    'on'    'on'    'text'
    'on'    'on'    'on'    'surface'
    'on'    'on'    'on'    'patch'
```

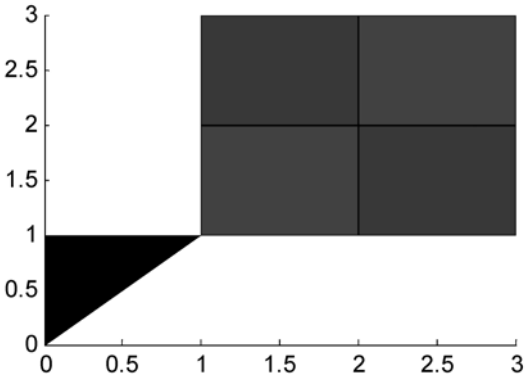


图 3-6　创建的属性效果

2．set 函数

功能：配置或显示设备对象属性。其调用格式如下：

set(obj)：显示 obj 的所有可配置属性。如果某个属性具有有限的字符串值列表，则这些值也将被显示出来。

props=set(obj)：将所有的可配置属性返回给 props。Props 为一结构数组，其字段为给定的属性名和可能的属性值的元素数组。如果属性没有有限的可能取值集合，则元素数组为空。

pv = set(h,'PropertyName')：返回 PropertyName 的有效值给 pv。pv 为一个可能值的元素数组，如果属性没有有限的可能值集合，则 pv 为空的元素数组。

set(H,'PropertyName',PropertyValue,...)：由一次声明来设置多个属性值。注意在同一次调用中，可以使用结构、属性名/属性值字符串对和属性名/属性值元素数组对。

set(obj, 'PropertyName')：显示 PropertyName 指定的属性有效值。PropertyName 必须具有一个有限的可能值的集合。

set(obj, S)：用结构中所包含的值来设置每个字段名中指定的属性，S 为一个结构，其字段名为设备对象的属性。

【例 3-24】为声卡创建模拟量输入对象 ai 并配置其工作于立体声模式。

```
>> ui=analoginput('winsound');
>> addchannel(ui,1:2)
Index:  ChannelName:  HwChannel:  InputRange:  SensorRange:  UnitsRange:  Units:
   1        'Left'         1          [-1 1]        [-1 1]        [-1 1]       'Volts'
   2        'Right'        2          [-1 1]        [-1 1]        [-1 1]
```

显示 ai 的所有可配置属性和它们的有效值。

```
>> set(ui)
        BufferingConfig
        BufferingMode: [ {Auto} | Manual ]
        Channel
        ChannelSkew
        ChannelSkewMode: [ {None} ]
        ClockSource: [ {Internal} ]
        DataMissedFcn: string -or- function handle -or- cell array
        InputOverRangeFcn: string -or- function handle -or- cell array
        InputType: [ {AC-Coupled} ]
        LogFileName
```

```
LoggingMode: [ Disk | {Memory} | Disk&Memory ]
LogToDiskMode: [ {Overwrite} | Index ]
ManualTriggerHwOn: [ {Start} | Trigger ]
Name
RuntimeErrorFcn: string -or- function handle -or- cell array
SampleRate
SamplesAcquiredFcn: string -or- function handle -or- cell array
SamplesAcquiredFcnCount
SamplesPerTrigger
StartFcn: string -or- function handle -or- cell array
StopFcn: string -or- function handle -or- cell array
Tag
Timeout
TimerFcn: string -or- function handle -or- cell array
TimerPeriod
TriggerChannel
TriggerCondition: [ {None} ]
TriggerConditionValue
TriggerDelay
TriggerDelayUnits: [ {Seconds} | Samples ]
TriggerFcn: string -or- function handle -or- cell array
TriggerRepeat
TriggerType: [ Manual | {Immediate} | Software ]
UserData
WINSOUND specific properties:
BitsPerSample
StandardSampleRates: [ {Off} | On ]
```

设置 SampleRate 属性的值为 10000。

```
>> set(ui,'SampleRate',10000)
```

3．setverify 函数

功能：配置并返回指定的属性。其调用格式如下：

Actual = setverify(obj,'PropertyName',PropertyValue)：将对象 obj 的 PropertyValue 设置为 PropertyName，并返回实际的属性值给 Actual。

Actual = setverify(obj.Channel(index),'PropertyName',PropertyValue)：将 index 指定的通道的 PropertyValue 设置为 PropertyName，并返回实际的属性值给 Actual。

Actual = setverify(obj.Line(index),'PropertyName',PropertyValue)：将 index 指定的数据线的 PropertyValue 设置为 PropertyName，并返回实际的属性值给 Actual。

【例 3-25】为 NI 板卡 AT-MIO-16DE-10 创建模拟量输入对象 ui，添加 8 个硬件通道，并使用 setverify 函数将采样速率设置为 10000Hz。

```
>> ui = analoginput('nidaq','Dev1');
ch = addchannel(ui,0:7);
ActualRate = setverify(ui,'SampleRate',10000);
%假定使用 setverify 函数将 ui 的所有通道的输入范围设置为-8V～8V
ActualInputRange = setverify(ai.Channel,'InputRange',[-8 8]);
%则 InputRange 的值被自动设置为-10V～10V。
```

```
ActualInputRange{1}
ans =
    -10      10
```

3.4.3 处理数据

在 MATLAB 中同时提供了一些函数实现对数据的处理，下面分别介绍如下。

1．flushdata 函数

功能：从数据获取引擎中卸载数据。其调用格式如下：

flushdata(GObj)：从数据采集引擎中卸载所有的数据，并将属性 SamplesAvailable 复位为 0。

flushdata(obj, 'mode')：以 mode 模式从数据采集引擎中卸载数据。

如果 mode 为 all，则 flushdata 将从引擎中卸载所有的数据并且属性 SamplesAvailable 被设置为 0。这种情况与 flushdata(obj)具有相同的效果。

如果 mode 为 triggers，则 flushdata 将在触发期间卸载获取的数据。triggers 的值仅在属性 TriggerRepeat 大于 0、属性 SamplePerTrigger 不为 inf 时有效。与最早的触发相关的数据将首先被卸载。

2．getdata 函数

功能：从数据获取引擎中提取数据、时间和事件信息。其调用格式如下：

data=getdata(obj)：对 obj 的每个通道，提取由属性 SamplesPerTrigger 所指定的样本数。data 是一个 m×n 的数组，其中 m 为要提取的样本数，n 为通道数。

data=getdata(obj, samples)：对 obj 的每个通道，提取由 samples 所指定的样本数。

data=getdata(obj, 'type')：提取指定格式的数据。如果 type 被设置为设备固有数据类型，则数据将以设备固有的类型形式返回。如果 type 被设置为双精度型，则每个通道的数据将以双精度形式返回。

data=getdata(obj, samples, 'type')：返回指定样本数的格式的数据。

[data, time]=getdata(…)：返回采样—时间对数据。Time 是一个 m×1 的相对时间数据数组。数组中的每个元素均为相对时间，单位为秒，表示了所对应的样本采样所需的时间，相对时间以引擎所注册的第一个样本为基准计算。

[data, time, abstime]=getdata(…)：提取样本—时间对，以及触发的绝对时间。绝对时间以时钟向量形式返回，并且与属性 InitialTriggerTime 中存储的值相同。

[data, time, abstime, events]=getdata(…)：提取样本—时间对，返回触发的绝对时间和一个包含事件列表的结构。所返回的事件与 EventLog 属性所存储的相同。

【例 3-26】设置一个数据记录任务，其配置为 60s。

```
>> da = opcda('localhost', 'Matrikon.OPC.Simulation');
connect(da);
grp = addgroup(da, 'ExOPCREAD');
itm1 = additem(grp, 'Triangle Waves.Real8');
itm2 = additem(grp, 'Saw-Toothed Waves.Int2');
set(grp, 'LoggingMode', 'memory', 'RecordsToAcquire', 60);
start(grp);
%从引擎中提取两个记录任务
>> s = getdata(grp, 2)
%检索到双数组中的所有剩余的数据并绘制
```

```
>> [itmID, val, qual, tStamp] = getdata(grp, 'double');
plot(tStamp(:,1), val(:,1), tStamp(:,2), val(:,2));
legend(itmID);
datetick x keeplimits
```

3．getsmaple 函数

功能：立即获取一个样本。其调用格式如下：

sample=getsample(obj)：立即返回包含对象 obj 的每个通道的一个样本的行向量。

【例 3-27】建模模拟量输入对象 ui 并添加 8 个通道。

```
>> ui = analoginput('nidaq','Dev1');
ch = addchannel(ui,0:7);
%返回每个通道中的一个样本
sample = getsample(ui);
```

3.4.4　获取信息和帮助

在 MATLAB 中同时也提供了一些函数实现获取信息和帮助。下面分别进行介绍。

1．daqhelp 函数

功能：显示设备对象、构造器、适配器、函数和属性的帮助信息。其调用格式如下：

daqhelp：显示整个数据采集工具箱中的构造器和函数的简洁描述信息。

out = daqhelp('name')：返回参数 name 所指定的设备对象、构造器、适配器、函数或属性的帮助信息，帮助文本返回到 out 中。

out = daqhelp(obj)：返回对象 obj 所包含的所有函数和属性的列表到输出参数 out。对象 obj 的构造器的帮助信息也被显示出来。

out = daqhelp(obj,'name')：返回设备对象 obj 的特定 name 的帮助信息。name 可以是构造器、适配器、属性和函数名。

可以通过在工作空间浏览器中右键单击设备对象，并在弹出的快捷菜单中选择【Explore】选项下的【DAQ Help】命令来显示帮助信息。

当显示属性帮助时，帮助文本的“参见段”中所有字母均大写的 name 是函数名，大小写字母均有 name 是属性名。

当显示函数帮助时，帮助文本的“参见段”中仅含有函数名。

【例 3-28】使用 daqhelp 函数来获取设备对象、构造器、适配器和属性的帮助信息的一些方法。

```
>> daqhelp('analogoutput');
out = daqhelp('analogoutput.m');
daqhelp set
daqhelp analoginput/peekdata
daqhelp analoginput.TriggerDelayUnits
```

运行程序，输出如下：

```
   Data Acquisition Toolbox
   Analog Output Functions and Properties.
   Data acquisition object construction.
      daq/analogoutput    - Construct analog output object.
   Getting and setting parameters.
      daqdevice/get         - Get value of data acquisition object property.
```

```
    daqdevice/set        - Set value of data acquisition object property.
    daqdevice/inspect - Open property inspector and configure data acquisition
                            object properties.
    setverify            - Set and return value of data acquisition object
                            property.
……
The trigger delay value is given by the TriggerDelay property.
The value of TriggerDelayUnits cannot be modified while the object is running.
See also TriggerDelay.
```

【例 3-29】使用 daqhelp 函数来显示设备对象 ui 的有关函数和属性的方法。

```
>> ui = analoginput('winsound');
daqhelp(ui,'InitialTriggerTime')
out = daqhelp(ui,'getsample');
```

运行程序，输出如下：

```
INITIALTRIGGERTIME double (read only)
InitialTriggerTime indicates the absolute time of the first trigger.
For all trigger types, InitialTriggerTime records the time when Logging or
Sending is set to On. The absolute time is recorded as a clock vector.
The InitialTriggerTime value can be returned with the GETDATA function,
and is also recorded in the Data.AbsTime field of the EventLog property.
See also EventLog, Logging, Sending, CLOCK, GETDATA.
```

2．daqhwinfo 函数

功能：显示数据采集硬件信息。其调用格式如下：

out = daqhwinfo：以结构的形式返回通用的硬件相关信息。所返回的信息包括安装的适配器、工具箱、MATLAB 版本以及工具箱名等。

out = daqhwinfo('adaptor')：返回特定的 adaptor 的相关硬件信息。所返回的信息包括适配器动态链接库名、硬件板卡名、ID 号以及设备对象构造器名等。

out = daqhwinfo('adaptor', 'Fieldname')：返回特定的 adaptor 的特定的 fieldname（字段名）的硬件相关信息。Fieldname 必须是单一的字符串。out 是一个数组元素。

out = daqhwinfo(obj)：返回设备对象 obj 的相关硬件信息。如果 obj 是一个设备对象数组，则 out 是一个 n 维数组，其中 n 为 obj 的长度。返回的信息取决于设备对象的类型，并且可以包含最大和最小采样率、通道获取、硬件通道、数据线 ID 以及代应商驱动器版本信息等。

out = daqhwinfo(obj,'FieldName')：返回设备对象 obj 的特定的 FieldName 的硬件相关信息。FieldName 可以是单一的字段名或字段名数组的单一元素。out 是一个 m×n 的数组，其中 m 是 obj 的长度，n 是 FieldName 的长度。

【例 3-30】daqhwinfo 应用。

```
%显示所安装的所有适配器（返回结果因用户的平台而各异）
>> out = daqhwinfo;
out.InstalledAdaptors
ans =
    'parallel'
    'winsound'
%显示所有安装的 winsound 设备的对象构造器名
>> out = daqhwinfo('winsound');
```

```
out.ObjectConstructorName
ans =
    [1x25 char]    [1x26 char]    ''
%创建声卡的模拟量输入对象 ui，显示 ui 的输入范围
>> ui = analoginput('winsound');
out = daqhwinfo(ui);
out.InputRanges
ans =
    -1    1
%显示 ui 的最大和最小采样速率
>> out = daqhwinfo(ui,{'MinSampleRate','MaxSampleRate'})
out =
    [5000]    [96000]
```

3．propinfo 函数

功能：为设备对象、通道或数据线返回属性特性。其调用格式如下：

out = propinfo(obj)：返回结构变量 out，其字段名为 obj 的属性名。out 中的每个属性名所包含的字段见表 3-1。

表 3-1　结构变量的属性字段

字 段 名	描　　述
Type	属性的数据类型。可取的值有 any, callback, double 和 string
Constrain	属性值的约束类型。可取的值有 bounded, callback, enum 和 none
ConstraintValue	属性值约束。约束可以是有效值范围或有效字符串值列表
DefaultValue	属性值的默认值
ReadOnly	如果属性为只读，则返回 1，否则返回 0
DeviceSpecific	如果属性为设备所特有的，则返回 1。如果返回值为 0，则表明此属性适合所有给定类型的设备对象

out = propinfo(obj,PropertyName)：返回 PropertyName 所指定的属性的结构变量 out。如果 PropertyName 为一个字符串数组的元素，则将返回一个结构变量数组，其每个元素对应每个属性。

【例 3-31】propinfo 函数应用。

```
% 为声卡创建模拟输入对象 ui，并将其配置为工作于立体声模式
>> ui=analoginput('winsound');
>> addchannel(ui,1:2);
% 获取 ui 的所有公共属性的全部属性信息
>> out=propinfo(ui)
out =
                BitsPerSample: [1x1 struct]
              BufferingConfig: [1x1 struct]
                BufferingMode: [1x1 struct]
                      Channel: [1x1 struct]
                 ChannelSkew: [1x1 struct]
             ChannelSkewMode: [1x1 struct]
                 ClockSource: [1x1 struct]
```

```
                DataMissedFcn: [1x1 struct]
                    EventLog: [1x1 struct]
           InitialTriggerTime: [1x1 struct]
            InputOverRangeFcn: [1x1 struct]
                    InputType: [1x1 struct]
                LogFileName: [1x1 struct]
                    Logging: [1x1 struct]
                LoggingMode: [1x1 struct]
              LogToDiskMode: [1x1 struct]
            ManualTriggerHwOn: [1x1 struct]
                        Name: [1x1 struct]
                    Running: [1x1 struct]
             RuntimeErrorFcn: [1x1 struct]
                  SampleRate: [1x1 struct]
             SamplesAcquired: [1x1 struct]
           SamplesAcquiredFcn: [1x1 struct]
      SamplesAcquiredFcnCount: [1x1 struct]
            SamplesAvailable: [1x1 struct]
           SamplesPerTrigger: [1x1 struct]
        StandardSampleRates: [1x1 struct]
                    StartFcn: [1x1 struct]
                     StopFcn: [1x1 struct]
                         Tag: [1x1 struct]
                    Timeout: [1x1 struct]
                   TimerFcn: [1x1 struct]
                 TimerPeriod: [1x1 struct]
              TriggerChannel: [1x1 struct]
           TriggerCondition: [1x1 struct]
      TriggerConditionValue: [1x1 struct]
                TriggerDelay: [1x1 struct]
           TriggerDelayUnits: [1x1 struct]
                  TriggerFcn: [1x1 struct]
              TriggerRepeat: [1x1 struct]
           TriggersExecuted: [1x1 struct]
                 TriggerType: [1x1 struct]
                        Type: [1x1 struct]
                    UserData: [1x1 struct]
%显示属性 SampleRateLoggingMode 的所有属性信息
>> out1 = propinfo(ai,'LoggingMode')
out1 =
                    Type: 'string'
              Constraint: 'enum'
         ConstraintValue: {'Disk'    'Memory'    'Disk&Memory'}
            DefaultValue: 'Memory'
                ReadOnly: 'whileRunning'
          DeviceSpecific: 0
```

3.4.5　综合应用

1．clear 函数

功能：将设备对象从 MATLAB 工作空间中清除。其调用格式如下：

clear obj：将对象 obj 及其所包含的所有相关通道或数据线从 MATLAB 工作空间中删除，但相关信息在数据采集引擎中仍然存在。

clear obj.Channel(index)：将对象 obj 所包含的相关通道从 MATLAB 工作空间中删除，但相关信息在数据采集引擎中仍然存在。

clear obj.Line(index)：将对象 obj 所包含的相关数据线从 MATLAB 工作空间中删除，但相关信息在数据采集引擎中仍然存在。

删除设备对象、通道和数据线操作应遵循如下规则：

clear 命令不能将对象、通道和数据线从数据采集引擎中删除。使用 delete 命令可以实现此目的。

如果设备对象的多个引用存在于工作空间中，清除某个引用并不会影响其他引用。

可以使用 daqfind 函数来还原从 MATLAB 工作空间中删除的设备对象。

2．daqfind 函数

功能：将设备对象、通道或数据线从数据采集引擎返回到工作空间。

out = daqfind：返回数据采集引擎中存在的所有设备对象。输出参数 out 是一个数组。

out = daqfind('PropertyName',PropertyValue,...)：返回数据采集引擎中符合所设定的属性名和属性值的所有设备对象、通道和数据线。属性名和属性值可以是向量数组。

out = daqfind(S)：返回数据采集引擎中符号 S 所指定的属性名和属性值的所有设备对象、通道或数据线。

out = daqfind(obj,'PropertyName',PropertyValue,...)：返回 obj 所指定的属性名和属性值的所有设备对象、通道或数据线。

daqfind 函数在下列环境中非常有用：

① 当设备对象已从 MATLAB 工作空间中删除，并需要将其从数据采集引擎中返回时。

② 当需要定位特定属性名和属性值的设备对象、通道或数据线时。

【例 3-32】daqfind 函数应用。

```
% 使用 daqfind 函数来返回已删除的设备对象
>> ui = analoginput('winsound');
ch = addchannel(ui,1:2);
set(ch,{'ChannelName'},{'Joe';'Jack'})
clear ui
ainew = daqfind;
% 返回通道名为 Lily 的通道
>> ch2 = daqfind(ainew,'ChannelName','Lily');
%返回采样速率为 8000Hz 设备对象描述名为 winsound0-AI 的设备对象
>> S.Name = 'winsound0-AI';
S.SampleRate = 8000;
daqobj = daqfind(S);
```

3．daqread 函数

功能：读取数据采集工具箱（.daq）文件。其调用格式如下：

data = daqread('filename')：从 filename 中读取所有的数据。data 为 m×n 数组数据矩阵。其中 m 为采样数，n 为通道数。

[data, time] = daqread(...)：返回采样—时间对。时间是一向量，其长度与数据相同，并且每个采样对应一个相对时间。相对时间以首次触发发生为计量基准。

[data, time, abstime] = daqread(...)：返回采样—时间对和首触发的绝对时间。abstime 以时钟向量形式返回。

[data, time, abstime, events] = daqread(...)：返回采样—时间对，首触发的绝对时间和事件日志。events 包含 Samples，Time 或 Triggers 属性值所设定的适当事件。

[data, time, abstime, events, daqinfo] = daqread(...)：返回采样—时间对，绝对时间，事件日志和 daqinfo 结构。Daqinfo 包含两个字段：ObjInfo 和 HwInfo。ObjInfo 是一个包含属性名/属性值对的结构，HwInfo 是一个包含硬件信息的结构。完整的事件日志将返回到 daqinfo，ObjInfo，Eventlog。

data = daqread('filename','Param1', Val1,...)：从 filename 中读取指定的数据。返回的数据的量和格式如表 3-2 中的属性设定。

表 3-2　属性说明

属　性　名	描　　述
Sample	设定采样范围
Time	设定相对时间范围
Triggers	设定触发器范围
Channels	设定通道范围。通道名可以作为数组元素指定
DataFormat	设定数据格式为 doubles 或 native
TimeFormat	设定时间格式为 vector 或 matrix

表 3-2 中的属性 Samples，Time 和 Triggers 是互斥的。

daqinfo = daqread('filename','info')：返回结构 daqinfo，daqinfo 包含两个字段：ObjInfo 和 HwInfo。ObjInfo 是一个包含属性名/属性值对的结构，HwInfo 是一个包含硬件信息的结构。完整的事件日志返回到 daqinfo.ObjInfo.EventLog。

【例 3-33】daqread 函数应用。

```
% 假定将 NI 板卡设置的模拟量输入对象 ui 配置如下，同时对象 ui 通过 4 个通道获取 1s 的数据，并将数据存储在输出文件 data.daq 中
>> ui = analoginput('nidaq','Dev1');
chans = addchannel(ui,0:3);
set(ui,'SampleRate',1000)
ActualRate = get(ui,'SampleRate');
set(ui,'SamplesPerTrigger, ActualRate')
set(ui,'LoggingMode','Disk&Memory')
set(ui,'LogFileName','data.daq')
start(ui)
% 数据存储到磁盘上后，可以通过函数 daqread 来获取文件中的数据和其他的与数据采%集相关的信息
>> [data,time] = daqread('data.daq');
%从 data.daq 中读取所有通道的 500～1000 的样本
```

```
>> data = daqread('data.daq','Samples',[500 1000]);
% 从 data.daq 中读取 1、2 的开始的 0.5s 的数据
[data, time] = daqread('data.daq', 'Triggers', [1 2]);
% 从 data.daq 中读取 2、4、7 通道的 1000～2000 的样本
>>data = daqread('data.daq', 'Samples', [1000 2000],...
                    'Channels', [2 4 7], 'DataFormat', 'native');
% 从 data.daq 中获取通道属性信息
>>daqinfo = daqread('data.daq','info');
chaninfo = daqinfo.ObjInfo.Channel;
% 获取 data.daq 文件所包含的事件类型表和事件数据
daqinfo = daqread('data.daq','info');
events = daqinfo.ObjInfo.EventLog;
event_type = {events.Type};
event_data = {events.Data};
%从 data.daq 读取所有数据和返回时间序列的集合对象
>> ata = daqread('data.daq','OutputFormat','tscollection');
```

4．delete 函数

功能：从数据采集引擎中卸载设备对象、通道或数据线。其调用格式如下：

delete(obj)：从引擎中卸载 obj 所指定的设备对象。如果 obj 包含有通道或数据线，也将被同时卸载。如果 obj 为最后访问驱动的设备对象，那么驱动和相关的适配器也将被卸载。

delete(obj.Channel(Index))：从引擎中卸载 obj 包含的由 index 指定的数据线。卸载后，剩余的通道应重新编号。

delete(obj.Line(Index))：从引擎中卸载 obj 包含的由 index 指定的数据线。卸载后，剩余的数据线应重新编号。

5．disp 函数

功能：显示设备对象、通道或数据线的概要信息。

disp(obj)：显示设备对象 obj 以及 obj 所包含的所有通道或数据线的概要信息。在命令行上直接输入 obj 也将显示相同的概要信息。

disp(obj.Channel(index))：显示对象 obj 中指定通道的概要信息。在命令行上直接输入 obj.Channel(index)也将显示相同的概要信息。

disp(obj.Line(index))：显示对象 obj 中指定的数据线的概要信息。在命令行上直接输入 obj.Line(index)也将显示相同的概要信息。

【例 3-34】disp 函数应用。

```
% 所有用于生产设备对象 AI 或 AI 所包含的通道的概要信息
>> AI=analoginput('winsound');
>> chans=addchannel(AI,1:2);
>> AI.SampleRate=44100;
>> AI.Channel(1).ChannelName='CH1';
>> chans
 Index:   ChannelName:   HwChannel:   InputRange:   SensorRange:   UnitsRange:   Units:
    1         'CH1'          1          [-1 1]         [-1 1]         [-1 1]      'Volts'
    2         'Right'        2          [-1 1]         [-1 1]         [-1 1]
```

6．length 函数

功能：返回设备对象、通道组或数据线组的长度。其调用格式如下：

out=length(obj)：将设备对象 obj 的长度返回给 out。

out=length(obj.Channel)：返回对象 obj 所包含的通道组的长度。

out=length(obj.Line)：返回对象 obj 所包含的数据线组的长度。

【例 3-35】length 函数应用。

```
% 为 NI 板卡创建模拟量输出对象 uo，并为其添加 8 个通道
>> ui=analoginput('nidaq',1);
>> uich=addchannel(ui,0:7);
%为 NI 板卡创建模拟输出对象 uo，为其添加一个通道，并创建设备对象数组 uiuo
>> uo=analogoutput('nidaq',1);
>> uoch=addchannel(uo,0);
>> uiuo=[ui uo]
Index :          Subsystem:          Name:
1                Analog Input        nidaq1-UI
2                Analog Output       nidaq1-UO
%实现返回 uiuo 的长度
length(uiuo)
ans =
     2
%实现返回模拟量输入通道组的长度
length(uich)
ans =
     8
```

7．muxchanidx 函数

功能：返回多路复用扫描型通道索引。其调用格式如下：

scanidx = muxchanidx(obj,muxboard,muxidx)：返回 muxidx 所指定的多路复用通道的扫描索引号。多路利用（mux）板由 muxboard 指定，对于每个 mux 板而言，其输入若为微分输入则 muxidx 取值范围是 0～31，若为单端输入则 muxidx 的取值范围是 0～63。muxboard 和 muxidx 为等长度的向量。

scanidx = muxchanidx(obj,absmuxidx)：返回 absmuxidx 所指定的多路复用通道的扫描索引号。absmuxidx 为 mux 板的单个通道的绝对索引号。

对于单端输入，第一块 mux 板的绝对索引值的取值范围为 0～63，第二块 mux 板的绝对索引值的取值范围为 64～127，第三块 mux 板的绝对索引值的取值范围为 128～191，第四块 mux 板的绝对索引值的取值范围为 192～255。例如，第四块 mux 板上的第二个单端输入通道（muxboard 等于 4，muxidx 等于 1）的索引值为 193。

【例 3-36】muxchanidx 函数应用。

```
% 为一个连接 4 个 AMUX-64T 多路复用器的 NI 板卡创建模拟量输入对象 ui，并使用函数 addmuxchannel 为 ui 添加 256 个通道
>> ui = analoginput('nidaq',1);
uai.InputType = 'SingleEnded';
ui.NumMuxBoards = 4;
addmuxchannel(ui);
%返回扫描索引值 14
>> scanidx = muxchanidx(ui,4,1);
scanidx = muxchanidx(ui,193);
```

8．size 函数

功能：返回设备对象、通道组或数据线组的大小。其调用格式如下：

d=size(obj)：返回二元行向量 d=[m, n]，此向量包含了 obj 的行数和列数。

[m1, m2, m3,…,mn]=size(obj)：将对象 obj 的前 n 维向量的长度返回给对应的变量。例如，[m, n]=size(obj)将行数返回给 m，列数返回给 n。

m=size(obj, dim)：返回由标量 dim 所指定的维数的长度。例如，size(obj, 1)返回行数。

d=size(obj.Channel)：返回二元行向量 d=[m, n]，d=[m, n]包含了通道组 obj.Channel 的行数和列数。

[m1, m2, m3,…,mn]=size(obj.Channel)：将通道组 obj.Channel 的前 n 维向量的长度返回给相应的输出变量。例如，[m, n]=size(obj.Channel)将行数返回给 m，将列数返回给 n。

m=size(obj.Channel, dim)：返回标量 dim 所指定的维数的长度。例如，size(obj.Channel, 1)返回行数。

d=size(obj.Line)：返回数据线组 obj.Line 所包含的行数和列数。

[m1, m2, m3,…,mn]=size(obj.Line)：将数据组组 obj.Line 前 n 维向量的长度返回给相应的输出变量。例如，[m, n]=size(obj.Line)返回行数给 m，返回列数给 n。

m=size(obj.Line, dim)：返回由标量 dim 所指定的维数的长度，例如，size(obj.Line, 1)返回行数。

【例 3-37】size 函数应用。

```
% 为 NI 板卡创建模拟量输入对象 ui，并添加 8 个通道
>> ui = analoginput('nidaq',1);
>> ch=addchannel(ui,0:7);
%返回设备对象的大小
>>size(ui)
ans =
1   1
% 返回通道组的大小
>> size(ch)
ans =
     8   1
```

第 4 章　信号的变换

4.1　Z 变换

4.1.1　Z 变换定义

一个序列 $x(n)$的 Z 变换 $X(z)$定义为

$$X(z)=Z[x(n)]=\sum_{n=-\infty}^{\infty}x(n)z^{-n} \tag{4-1}$$

式中：z 为复变量。又可写为：$X(z)=z[x(n)]$。

傅里叶变换级数的收敛条件是要求序列 $x(n)$绝对可和。而 $X(z)=\sum_{n=-\infty}^{\infty}x(n)z^{-n}$ 也是一幂级数，只有收敛时 Z 变换才有意义，因此要求：

$$\left|X(z)\right|=\left|\sum_{n=-\infty}^{\infty}x(n)z^{-n}\right|\leqslant\sum_{n=-\infty}^{\infty}x(n)\,|\,z\,|^{-n}\leqslant\infty \tag{4-2}$$

使幂级数收敛的全部 z 的集合称为 Z 变换 $X(z)$的收敛域，即只有 z 在收敛域内取值，Z 变换才有意义。

从收敛公式中可以看出，由于序列 $x(n)$和$|z|^{-1}$ 相乘，这就使得一个序列可能不是绝对可和的，其傅里叶变换不存在，而它的 Z 变换却可能存在。$X(z)$是否收敛，取决于 $x(n)$，不同的 $x(n)$对应不同的 Z 变换收敛域。下面直接给出不同形式的序列所对应的 Z 变换收敛域。

4.1.2　Z 变换的性质

1．线性特性

若两序列 $x(n)$和 $y(n)$的 Z 变换为

$$\begin{aligned}&Z[x(n)]=X(z)\quad(R_{x^-}<|\,z\,|<R_{x^+})\\&Z[y(n)]=Y(z)\quad(R_{y^-}<|\,z\,|<R_{y^+})\end{aligned} \tag{4-3}$$

则其线性组合 $ax(n)+by(n)$的 Z 变换为

$$Z[ax(n)+b[y(n)]]=aX(z)+bY(z) \tag{4-4}$$

式中：$R_-<|z|<R_+$；$R_-=\max[R_{x^-}, R_{y^-}]$；$R_+=\min[R_{x^+}, R_{y^+}]$。

2．时域平移性

1）双边 ZT

（1）左移：

$$Z_B[x(n+m)]=z^mX(z)\quad(R_1<|\,z\,|<R_2) \tag{4-5}$$

（2）右移：

$$Z_B[x(n-m)]=z^{-m}X(z)\quad(R_1<|\,z\,|<R_2) \tag{4-6}$$

（3）序列时移最多只会使 ZT 在 z=0 或 z=∞处的零点、极点情况发生变化。

2）*单边序列*

对因果序列：

$$Z[x(n-m)u(n)]=z^{-m}X(z) \tag{4-7}$$

3．时域扩展性

定义

$$x_{(a)}(n)\triangleq\begin{cases}x\left(\dfrac{n}{a}\right), & \dfrac{n}{a}\in Z\\ 0, & \dfrac{n}{a}\notin Z\end{cases}\quad(0\neq a\in Z) \tag{4-8}$$

式中：a 为扩展因子。

如序列是偶对称的，则

$$X(z)=Z[x(n)]=Z[x(-n)]=X\left(\frac{1}{z}\right) \tag{4-9}$$

如果一个偶对称或奇对称序列的 ZT 含有一个非零的零点（或极点）z_0，那么它必含有另外一个与之互为倒数的零点（或极点）$\dfrac{1}{z_0}$。

4．*z* 域尺度变换（或序列指数加权）性

定义　$$Z[a^n x(n)]=X\left(\frac{z}{a}\right)\quad\left(R_{x1}<\left|\frac{z}{a}\right|<R_{x2}\right) \tag{4-10}$$

5．*z* 域微分（或序列线性加权）性

定义　$$Z[nx(n)]=-z\frac{\mathrm{d}}{\mathrm{d}z}Z[x(n)]\quad(R_1<|z|<R_2) \tag{4-11}$$

4.1.3　Z 反变换

X(*z*)的反变换定义为

$$x(n)=Z^{-1}[X(z)]=\frac{1}{2\pi\mathrm{j}}\oint X(z)z^{n-1}\mathrm{d}z \tag{4-12}$$

求 Z 反变换的方法通常有三种：围线积分法（留数法）、幂级数展开法（长除法）及部分分式展开法。

注意：$x(n)$与 $X(z)$并不是一一对应的，即相同的 $X(z)$表达式会因给出的收敛域不同而各自对应着不同的 $x(n)$，比如：$\dfrac{z}{z-a}$ 当收敛域为$|z|>a$ 时，其反变换为 $a^n u(-n-1)$。

在 MATLAB 中，利用留数法求解 Z 变换可以通过函数 residuez 来实现，其调用格式如下：

[r,p,k] = residuez(b,a)：参数 b，a 分别为有理 Z 函数的分子多项式的系数向量与分母多项式的系数向量，参数 r 与 p 为列向量，r 为留数，p 为极点，如果分子多项式的阶数大于分母多项式的阶数，参数 K 返回直接项系数。

4.1.4 Z 变换的 MATLAB 实现

假设一个 LSI 系统，其差分方程可表示为

$$y(n)=-\sum_{k=1}^{N}a(k)y(n-k)+\sum_{r=0}^{M}b(r)x(n-r) \tag{4-13}$$

给定 $x(n)$及 $y(n)$的初始条件，求出系统的输出 $y(n)$的表达式，这是差分方程的求解问题。

1．系统的零输入解

系统输入为零，即 $x(n)=0$ 时，有

$$y(n)+\sum_{k=1}^{N}a(k)y(n-k)=0 \tag{4-14}$$

此时，方程的解由 $y(n)$的初始条件引起。对该式两边取 Z 变换，并令 $a(0)=1$，则

$$Y(z)=\frac{-\sum_{k=0}^{N}a(k)z^{-k}\left[\sum_{k=0}^{-1}y(m)z^{-m}\right]}{\sum_{k=0}^{N}a(k)z^{-k}} \tag{4-15}$$

取 Z 逆变换，即得到系统的零输入解为

$$y_{0i}=Z^{-1}[Y(z)] \tag{4-16}$$

2．系统的零状态解

系统输出 $y(n)$的初始状态为零，且 $x(n)$是因果序列，那么 Z 变换为

$$Y(z)=\frac{\sum_{r=0}^{M}b(r)z^{-r}}{1+\sum_{k=0}^{N}a(k)z^{-k}}=H(z)X(z) \tag{4-17}$$

由此得到的 $y(n)$称为零状态解，它是单纯由输入引起的输出，即

$$y_{0s}=Z^{-1}[H(z)X(z)] \tag{4-18}$$

系统完整的输出应是零状态解与零输入解之和，即

$$y(n)=y_{0i}(n)+y_{0s}(n) \tag{4-19}$$

在 MATLAB 中提供了 filtic 函数实现 Z 变换。其调用格式如下：

z = filtic(b,a,y,x)

z = filtic(b,a,y)

【例 4-1】系统的微分方程为

$$y(n)-2y(n-1)+3y(n-2)=4u(n)-5u(n-1)+6u(n-2)-7u(n-3)$$

其初始条件为 $x(-1)=1$，$x(-2)=-1$，$y(-1)=-1$，$y(-2)=1$，求系统的输出 $y(n)$。

实现的 MATLAB 程序代码如下：

```
>> clear all;
b=[4 -5 6 -7];
a=[1 -2 3];
x0=[1 -1];
y0=[-1 1];
xic=filtic(b,a,y0,x0)
bxplus=1;
```

```
axplus=[1 -1];
ayplus=conv(a,axplus)
byplus=conv(b,bxplus)+conv(xic,axplus)
[R,P,K]=residuez(byplus,ayplus)
Mp=abs(P)
Ap=angle(P)*180/pi
N=100;
n=0:N-1;
xn=ones(1,N);
yn=filter(b,a,xn,xic);
plot(n,yn);
```

运行程序，输出如下，效果如图 4-1 所示。

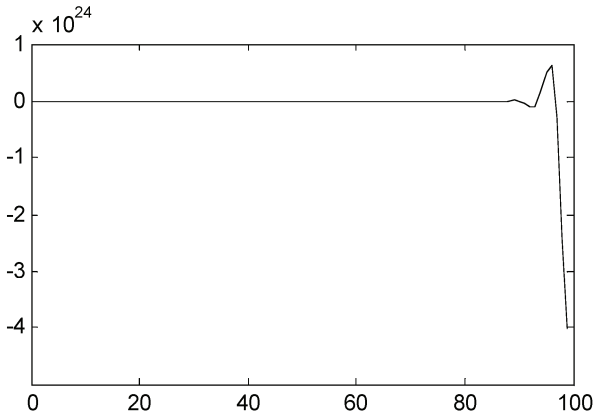


图 4-1　Z 变换的系统输出响应曲线

```
xic =
    -16     16     -7
ayplus =
     1     -3      5     -3
byplus =
    -12     27    -17      0
R =
  -5.5000 - 1.0607i
  -5.5000 + 1.0607i
  -1.0000
P =
   1.0000 + 1.4142i
   1.0000 - 1.4142i
   1.0000
K =
     0
Mp =
    1.7321
    1.7321
    1.0000
Ap =
```

```
   54.7356
  -54.7356
         0
```

4.2 离散傅里叶变换

离散傅里叶变换（DFT）是数字信号处理和分析的重要工具和方法之一。为了更好地理解 DFT 的概念，下面从周期序列和离散傅里叶级数开始进行说明。

4.2.1 周期序列和傅里叶级数

对于一个周期序列，可以表示如下：

$$\tilde{x}(n)=\tilde{x}(KN+n) \tag{4-20}$$

式中：K 为任意整数；N 为信号的周期。

由于周期序列不是绝对可加的，不能进行 Z 变换，但周期序列可以用离散傅里叶级数来表示，即

$$\tilde{X}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}kn}\quad(n=0,\pm1,\pm2,\cdots) \tag{4-21}$$

式中：$\tilde{X}(k)$ 为离散傅里叶级数的系数，并且

$$\tilde{X}(k)=\sum_{n=0}^{N-1}\tilde{x}(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}nk}\quad(k=0,\pm1,\pm2,\cdots) \tag{4-22}$$

为表示方便，令 $W_N=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$，因此，周期序列的离散傅里叶级数变换对可以表示为

$$\begin{cases}\tilde{X}(k)=\sum\limits_{n=0}^{N-1}\tilde{x}(n)W_N^{kn}\\ \tilde{x}(n)=\dfrac{1}{N}\sum\limits_{k=0}^{N-1}\tilde{X}(k)W_N^{-kn}\end{cases} \tag{4-23}$$

周期序列 $\tilde{X}(k)$ 可看成是对 $\tilde{x}(n)$ 的一个周期 $x(n)$作 Z 变换，然后在 Z 平面的单位圆上，按照等间隔进行采样而得到，即

$$\tilde{X}(k)=X(z)\,|\,z=W_N^{-k}=\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}k} \tag{4-24}$$

4.2.2 离散傅里叶变换介绍

前面已经介绍，周期序列 $\tilde{x}(n)$ 的离散傅里叶级数的系数 $\tilde{X}(k)$ 也是周期性的。为了保持时域和频域之间的对偶关系，选择周期序列 $\tilde{x}(n)$ 的一个周期 $x(n)$和离散傅里叶级数的系数 $\tilde{X}(k)$ 相对应的一个周期 $X(k)$，构成一个离散傅里叶变换对，称为有限长序列的离散傅里叶变换对，即

$$\begin{cases}X(k)=\mathrm{DFT}[x(n)]=\sum\limits_{n=0}^{N-1}x(n)W_N^{kn} & (0\leqslant k\leqslant N-1)\\ x(n)=\mathrm{IDFT}[X(k)]=\dfrac{1}{N}\sum\limits_{n=0}^{N-1}X(k)W_N^{-kn} & (0\leqslant n\leqslant N-1)\end{cases} \tag{4-25}$$

式中：$W_N = e^{-j\frac{2\pi}{N}}$，称为旋转因子。

显然，在以矩阵为基础的 MATLAB 中，根据以下关系，可以很容易实现 DFT 运算。

$$[X(0) \quad X(1) \quad X(2) \quad \cdots \quad X(-1)]$$

$$=[x(n)]*[W_N]=[x(0) \quad x(1) \quad xX(2) \quad \cdots \quad xX(-1)]\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix}$$

式中：DFT 矩阵（旋转因子矩阵 $\boldsymbol{W}$）可以由信号处理工具箱中的 dftmtx 函数实现。其调用格式如下：

```
A = dftmtx(n)
```

函数调用后，返回一个 $n \times n$ 的 DFT 的旋转因子矩阵 $\boldsymbol{W}$。这样，对给定长度为 n 的行向量信号 $x(n)$，利用语句：Xk=k*W 就可以获得 $x(n)$的傅里叶变换 $X(k)$。反之，用 MATLAB 实现 IDFT 也非常简单，只需计算出离散傅里叶反变换（IDFT）的旋转因子矩阵为 $\boldsymbol{A_i}$ 即可。其调用格式如下：

```
Ai = conj(dftmtx(n))/n
```

【例 4-2】利用 MATLAB 计算序列 $x(n)$的 DFT。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=linspace(1e-3,100e-3,10);
xn=sin(100*2*pi*t);              %产生有限序列 x(n)
N=length(xn);                    %获得序列的长度
WNnk=dftmtx(N);
Xk=xn*WNnk;                      %计算 x(n)的 DFT
subplot(1,2,1);stem(1:N,xn);
xlabel('(a)  时域离散序列 x(n)');
subplot(1,2,2);stem(1:N,abs(Xk));
xlabel('(b) x(n)的 DFT 变换结果');
```

运行程序，输出如下，效果如图 4-2 所示。

```
Xk =
  Columns 1 through 4
   0.0000                  2.9389 - 4.0451i   -0.0000 + 0.0000i   -0.0000 - 0.0000i
  Columns 5 through 8
   0.0000 + 0.0000i   -0.0000                  0.0000 - 0.0000i   -0.0000 + 0.0000i
  Columns 9 through 10
  -0.0000 - 0.0000i    2.9389 + 4.0451i
```

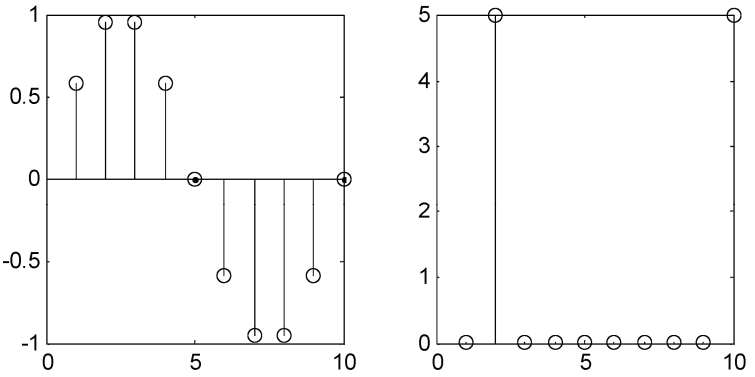


图 4-2　有限序列及其 DFT 效果

对于 IDFT，利用同样的方法可以非常容易地实现。

【例 4-3】实现 IDFT。

其实现的 MATLAB 程序代码如下：

```
>> N=length(Xk);                %获得序列的长度
n=0:N-1;
k=0:N-1;
WN=exp(-j*2*pi/N);
WNnk=WN.^(-n'*k);              %计算旋转因子矩阵
xn=Xk*WNnk/N;                  %计算 X(k)的 IDFT
subplot(1,2,1);stem(1:N,Xk);
xlabel('(a) X(k)');
subplot(1,2,2);stem(real(xn));
xlabel('(b) X(k)的 IDFT 变换结果');
```

运行程序，效果如图 4-3 所示。

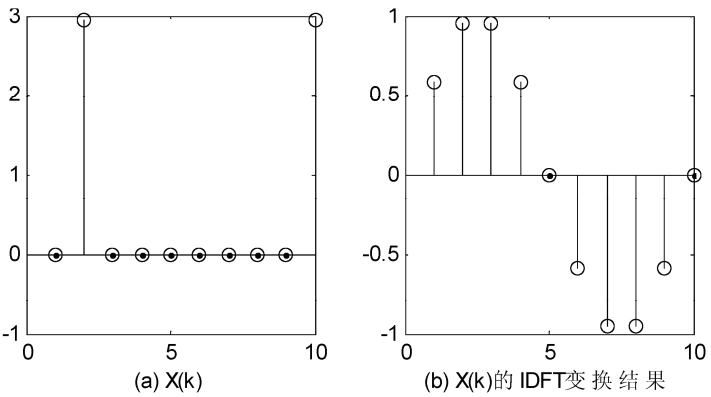


图 4-3　IDFT 变换效果

4.2.3　离散傅里叶变换的性质

同其他变换一样，有限长序列的离散傅里叶变换也有许多重要的性质。

1．线性性质

如果 $x_1(n)$和 $x_2(n)$分别是两个有限长序列，长度分别为 N_1 和 N_2，其 DFT 分别为 $X_1(k)$和 $X_2(k)$，

则有

$$\mathrm{DFT}[ax_1(n)+bx_2(n)]=a\,\mathrm{DFT}[x_1(n)]+b\,\mathrm{DFT}[x_2(n)]=aX_1(k)+bX_2(k) \qquad (4\text{-}26)$$

【例 4-4】设有两个振动 cos(2*pi*0.24*t)和 cos(2*pi*0.12*t)。求两个振动及其合成振动的离散傅里叶变换，采样间隔为 1s，数据长度为 100。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
N=100;dt=1;                                  %数据长度为 100,采样间隔为 1s
n=0:N-1;t=n*dt;                              %给出时间序列
xn1=cos(2*pi*0.24*t);                        %第一个振动
xn2=cos(2*pi*0.26*t);                        %第二个振动
Xk1=dfs(xn1,N);                              %第一个振动的傅里叶变换
Xk2=dfs(xn2,N);                              %第二个振动的傅里叶变换
magXk1=abs(Xk1);                             %第一个振动的的振幅
phaXk1=angle(Xk1);                           %第一个振动的相位
k=0:length(magXk1)-1;
subplot(311);plot(k/(N*dt),magXk1*2/N);      %绘制第一个振动的振幅谱
ylabel('振幅');title('第一个振动的傅里叶变换');
magXk2=abs(Xk2);                             %第二个振动的的振幅
phaXk2=angle(Xk2);                           %第二个振动的相位
k=0:length(magXk2)-1;
subplot(312);plot(k/(N*dt),magXk2*2/N);      %绘制第二个振动的振幅谱
ylabel('振幅');title('第二个振动的傅里叶变换');
Xk=dfs(xn1+xn2,N);                           %两个振动合成的傅里叶变换
magXk=abs(Xk);                               %合成振动的振幅
phaXk=angle(Xk);                             %合成振动的相位
k=0:length(magXk2)-1;
subplot(313);plot(k/(N*dt),magXk*2/N);       %绘制两个合成振动的振幅和相位
ylabel('振幅');title('合成的振动的傅里叶变换');
```

运行该程序效果如图 4-4 所示。可以看出，两个振动的傅里叶变换的叠加在频域轴上与合成振动的傅里叶变换是一致的，清楚地表明傅里叶变换的线性。

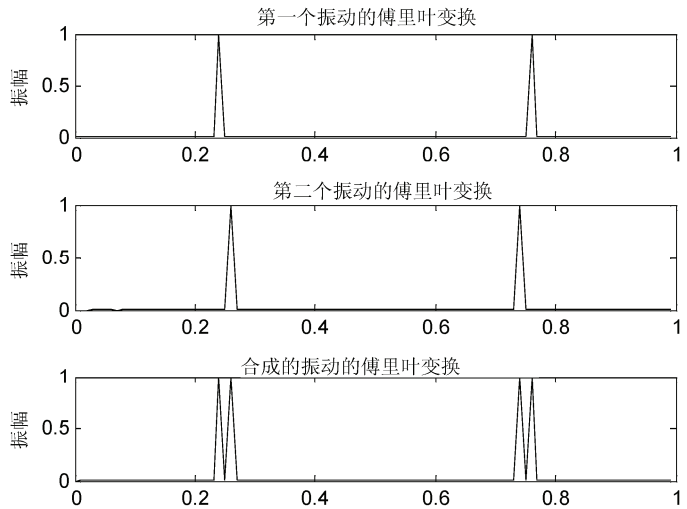


图 4-4　0.12Hz 和 0.24Hz 的两个振动及其合成振动的傅里叶变换图示

2．循环移位

设 $x(n)$为有限长序列，长度为 N，则 $x(n)$的循环移位 $f(n)=x((n+m))_N R_N(n)$，其中 $x((n+m))_N$ 称为序列 $x(n)$的周期延拓。则序列 $x(n)$移位后的序列 $f(n)$的 DFT 为

$$\mathrm{DFT}[f(n)]=W_N^{-kn}X(k) \tag{4-27}$$

其中：$X(k)$为 $x(n)$的 DFT。

由于时域和频域之间存在对偶关系，所以当序列 $x(n)$的 DFT 系数作循环移位时，与 $x(n)$的特性类似，即已知

$$\begin{aligned} X(k)&=\mathrm{DFT}[x(n)] \quad (0\leqslant k\leqslant N-1)\\ Y(k)&=X((k+L))_N R_N(n)\end{aligned} \tag{4-28}$$

则必有

$$y(n)=\mathrm{IDFT}[Y(k)]=W_N^{nL}x(n) \tag{4-29}$$

利用 MATLAB 提供的 mod 函数，根据关系式 $f(n)=x(\mathrm{mod}((m+n),N)+1)$，可以较容易地实现序列移位的 DFT 变换和 IDFT 变换。

【例 4-5】设 $x(n)=3e^n$，$0\leqslant n\leqslant 15$，求 $f(n)=x((n+m))_{15}R_{15}(n)$ 的 DFT 及其 IDFT。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=15;                          %有限长序列的长度
n=0:N-1;
xn=3*exp(n);                   %移位前序列
m=5;                           %移动长度,左移为正数,右移为负数
fn=xn(mod((n+m),N)+1);         %移位后序列
XK=DFT(xn);                    %将前面的 DFT 程序变换成子程序,这里直接调用
>> FK=IDFT(fn)                 %移位后序列 f(n)的 IDFT
```

运行程序，输出如下，效果如图 4-5 所示。

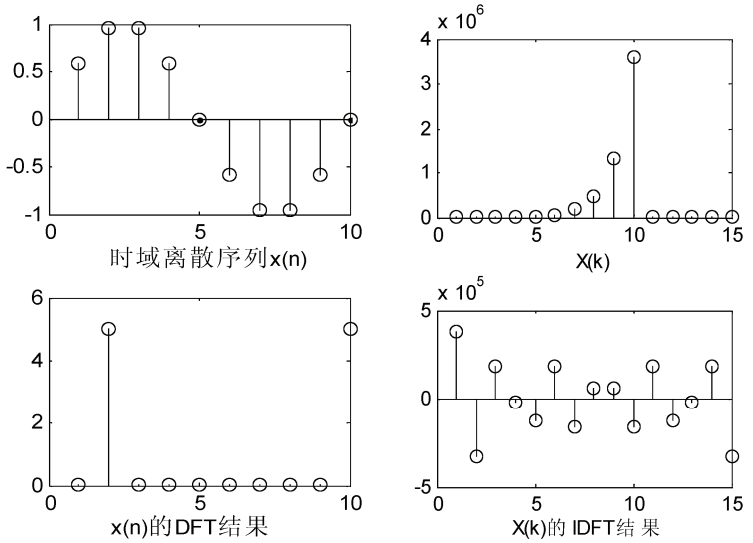


图 4-5　两种序列的效果

```
Xk =
Columns 1 through 4
    0.0000              2.9389-4.0451i      -0.0000+0.0000i      -0.0000-0.0000i
```

```
Columns 5 through 8
   0.0000+0.0000i   -0.0000                  0.0000-0.0000i        -0.0000+0.0000i
Columns 9 through 10
   -0.0000-0.0000i   2.9389+4.0451i
FK =
  1.0e+005 *
Columns 1 through 5
   3.8050          -3.2459-1.3978i   1.8439+2.3657i   -0.1559-2.5193i   -1.2402+1.7983i
Columns 6 through 10
   1.8944-0.5098i  -1.6357-0.8169i   0.6392+1.6462i   0.6392-1.6462i    -1.6357+0.8169i
Columns 11 through 15
   1.8944+0.5098i  -1.2402-1.7983i  -0.1559+2.5193i   1.8439-2.3657i    -3.2459+1.3978i
```

3．循环卷积

设有限长序列 $x_1(n)$和 $x_2(n)$，长度分别为 N_1 和 N_2，它们的 DFT 分别为

$$\begin{cases} X_1(k)=\mathrm{DFT}[x_1(n)], & 0\leqslant k\leqslant N-1 \\ X_2(k)=\mathrm{DFT}[x_2(n)], & 0\leqslant k\leqslant N-1 \end{cases} \tag{4-30}$$

取 $N=\max\{N_1,N_2\}$，如果 $f(n)=x_1(n)\otimes x_2(n)$,$0\leqslant n\leqslant N-1$（⊗ 表示循环卷积），则

$$F(k)=\mathrm{DFT}[f(n)]=X_1(k)X_2(k),0\leqslant k\leqslant N-1 \tag{4-31}$$

而如果：$f(n)=x_1(n)\cdot x_2(n),0\leqslant k\leqslant N-1$，则

$$\begin{aligned} F(k)&=\mathrm{DFT}[f(n)]=\frac{1}{N}X_1(k)\otimes X_2(k) \\ &=\frac{1}{N}\sum_{l=0}^{N-1}X_2(k)X_1((k-l))_N R_N(k), \quad 0\leqslant k\leqslant N-1 \end{aligned} \tag{4-32}$$

根据循环卷积的定义，用户可自定义编写构造循环卷积函数 circonvt，其源代码如下：

```
function fn=circonvt(x1,x2,N)
% circonvt 函数实现输入序列 x1 和 x2 的循环卷积,fn 为输出序列
% N 为循环卷积长度
%  实现方法:fn=sum(x1(m)*x2((n-m) mod N))
if (length(x1)>N |length(x2)>N)                    %判断输入信号的长度
    error('N 的长度必须大于输入数据的长度');
end
x1=[x1,zeros(1,N-length(x1))];
x2=[x2,zeros(1,N-length(x2))];
m=0:N-1;
x=zeros(N,N);
for n=0:N-1
    x(:,n+1)=x2(mod((n-m),N)+1)';
end
fn=x1*x;
```

【例 4-6】计算序列 $x_1(n)=2e^{n1/5}(0\leqslant n\leqslant 15)$和 $x_2(n)=5(0.3)^{n2/3}(0\leqslant n\leqslant 18)$的循环卷积，并计算其循环卷积的 IDFT。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n1=0:15;
x1=2*exp(n1/5)
```

```
n2=0:18;
x2=5*(0.3).^(n2/3)
N=20;
fn=circonvt(x1,x2,N)
Fk=IDFT(fn)
figure;
subplot(2,2,1);stem(n1,x1);
xlabel('(a) x1');
subplot(2,2,2);stem(n2,x2);
xlabel('(b) x2');
subplot(2,2,3);stem(0:N-1,fn);
xlabel('(c) x1 和 x2 循环卷积结果 fn');
subplot(2,2,4);stem(0:N-1,Fk);
xlabel('(d) fn 的 DFT 结果');
```

运行程序，输出如下，效果如图 4-6 所示。

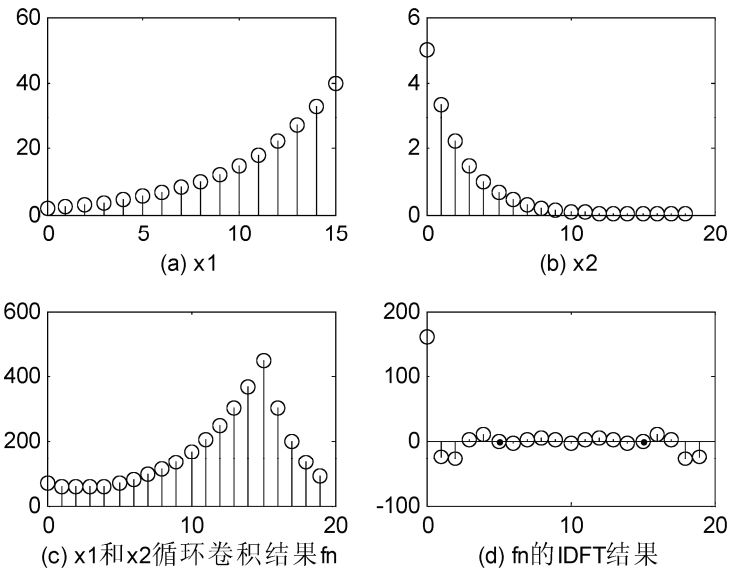


图 4-6　序列 $x_1(n)$和 $x_2(n)$循环卷积及其 IDFT 效果

```
x1 =
   Columns 1 through 8
    2.0000    2.4428    2.9836    3.6442    4.4511    5.4366    6.6402    8.1104
   Columns 9 through 16
    9.9061   12.0993   14.7781   18.0500   22.0464   26.9275   32.8893   40.1711
x2 =
   Columns 1 through 8
    5.0000    3.3472    2.2407    1.5000    1.0041    0.6722    0.4500    0.3012
   Columns 8 through 16
    0.2017    0.1350    0.0904    0.0605    0.0405    0.0271    0.0181    0.0121
Columns 17 through 19
    0.0081    0.0054    0.0036
fn =
Columns 1 through 8
```

```
    69.7402   58.8931   54.3344   54.5835   58.7822   66.5173   77.7103   92.5496
Columns 9 through 16
   111.4566  135.0731  164.2689  200.1634  244.1620  298.0072  363.8443  444.4247
Columns 17 through 20
   297.5126  199.1647  133.3274   89.2489
Fk =
  1.0e+002 *
Columns 1 through 4
    1.6069            -0.2576 - 0.6571i   -0.2813 + 0.0786i    0.0232 + 0.1498i
Columns 5 through 8
 0.0900 - 0.0081i   -0.0059 - 0.0617i    -0.0490 + 0.0054i    0.0031 + 0.0427i
Columns 9 through 12
 0.0377 - 0.0003i    0.0008 - 0.0335i    -0.0317 + 0.0000i    0.0008 + 0.0335i
Columns 13 through 16
0.0377 + 0.0003i    0.0031 - 0.0427i    -0.0490 - 0.0054i    -0.0059 + 0.0617i
Columns 17 through 20
0.0900 + 0.0081i    0.0232 - 0.1498i    -0.2813 - 0.0786i    -0.2576 + 0.6571i
```

4．共轭对称性

设 $x^*(n)$是序列 $x(n)$的复共轭序列，长度为 N，则

$$\mathrm{DFT}[x^*(n)]=X^*((-k))_N R_N(k) \tag{4-33}$$

如果用 $x_r(n)$和 $x_i(n)$表示序列 $x(n)$的实部和虚部，即

$$x(n)=x_r(n)+\mathrm{j}x_i(n) \tag{4-34}$$

其中

$$\begin{cases} x_r(n)=\mathrm{Re}[x(n)]=\dfrac{1}{2}[x(n)+x^*(n)] \\ \mathrm{j}x_i(n)=\mathrm{j}\,\mathrm{Im}[x(n)]=\dfrac{1}{2}[x(n)-x^*(n)] \end{cases} \tag{4-35}$$

此外，序列 $x(n)$也可以用共轭对称序列 $x_e(n)$和共轭反对称序列 $x_o(n)$表示，即

$$x(n)=x_e(n)+x_o(n) \tag{4-36}$$

其中

$$\begin{cases} x_e(n)=\dfrac{1}{2}[x(n)+x^*(-n)] \\ x_o(n)=\dfrac{1}{2}[x(n)-x^*(-n)] \end{cases} \tag{4-37}$$

经证明，可得

$$\begin{cases} \mathrm{DFT}[x_r(n)]=X_e(k)=\dfrac{1}{2}[X(k)+X^*(-k)] \\ \mathrm{DFT}[\mathrm{j}x_i(n)]=X_o(k)=\dfrac{1}{2}[X(k)-X^*(-k) \end{cases} \tag{4-38}$$

$$\begin{cases} \mathrm{DFT}[x_e(n)]=\mathrm{Re}[X(k)]=\dfrac{1}{2}[X(k)+X^*(k)] \\ \mathrm{DFT}[x_o(n)]=\mathrm{j}\,\mathrm{Im}[X(k)]=\dfrac{1}{2}[X(k)-X^*(k)] \end{cases} \tag{4-39}$$

【例 4-7】设 $x(n)=2e^{((-n)/3+j2n)}(0\leqslant n\leqslant 10)$，验证：DFT[$x_e(n)$]=Re[$X(k)$]，DFT[$x_o(n)$]=jIm[$X(k)$]。其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=10;
n=0:N-1;
xn=2*exp(-n/3+j*2*n);
xn1=xn(mod(N-n,N)+1);
Rexn1=real(xn1);
Imxn1=imag(xn1);
xn1=Rexn1-j*Imxn1;                    %计算 x*(-n)
xne=(xn+xn1)/2;                       %计算 xe(n)
xno=(xn-xn1)/2;                       %计算 xo(n)
ReXk=real(DFT(xn));
ImXk=imag(DFT(xn));
Xek=DFT(xne);
Xok=DFT(xno);
figure;
subplot(4,2,1);stem(n,xn);
xlabel('(a) x(n)');
subplot(4,2,2);stem(n,xn1);
xlabel('(b) x*(-n)');
subplot(4,2,3);stem(n,xne);
xlabel('(c) xe(n)');
subplot(4,2,4);stem(n,xno);
xlabel('(d) xo(n)');
subplot(4,2,5);stem(0:N-1,real(Xek));
xlabel('(e) xe(n)的 DFT 结果');
subplot(4,2,6);stem(0:N-1,imag(Xok));
xlabel('(f) xo(n)的 DFT 结果');
subplot(4,2,7);stem(0:N-1,ReXk);
xlabel('(j) x(n)的 DFT 的实部');
subplot(4,2,8);stem(0:N-1,ImXk);
xlabel('(k) x(n)的 DFT 的虚部');
```

运行程序，输出如下，效果如图 4-7 所示。

```
ReXk =
Columns 1 through 8
    0.1233    0.1167    0.1147    0.1160    0.1227    0.1462    0.2708    0.6383
Columns 9 through 10
   0.2100    0.1412
ImXk =
Columns 1 through 8
   0.0569    0.0213   -0.0093   -0.0411   -0.0812   -0.1450   -0.2709    0.1596
Columns 9 through 10
   0.2018    0.1080
Xek =
Columns 1 through 5
 0.1233             1167 + 0.0000i    0.1147             0.1160 - 0.0000i    0.1227 - 0.0000i
```

```
Columns 6 through 10
0.1462 - 0.0000i   0.2708 - 0.0000i   0.6383 - 0.0000i   0.2100 + 0.0000i   0.1412 + 0.0000i
Xok =
Columns 1 through 4
   0.0000 + 0.0569i      -0.0000 + 0.0213i      -0.0000 - 0.0093i      -0.0000 - 0.0411i
Columns 5 through 8
   -0.0000 - 0.0812i      -0.0000 - 0.1450i       0.0000 - 0.2709i       0.0000 + 0.1596i
Columns 9 through 10
    0.0000 + 0.2018i      -0.0000 + 0.1080i
```

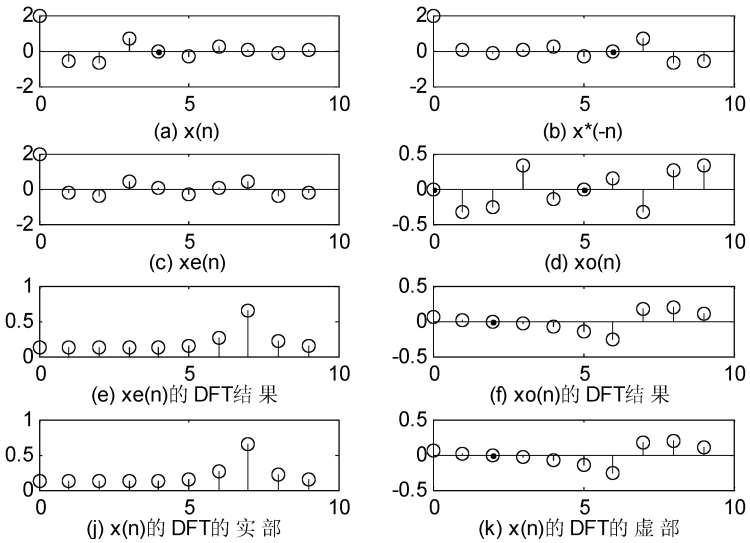


图 4-7　序列 xe(n)和 xo(n)的 DFT 与 x(n)的 DFT 之间的关系

4.2.4　离散傅里叶变换参数对频率分辨率的影响

频率分辨率就是在频率轴上能分辨出的两个频率点的最小间隔，即

$$\Delta f = \frac{f_{\max} - f_{\min}}{N} \tag{4-40}$$

式中：$f_{\max}$ 为该信号的最大频率分量；$f_{\min}$ 为该信号的最小频率分量；N 为频域有效数据长度。

要能有效地区分频率轴上的两个频率点 f_1 和 f_2，有效数据长度 N 必须满足以下关系式：

$$\frac{2f_s}{N} < | f_1 - f_2 | \tag{4-41}$$

可以对序列进行补零操作，以增加数据的长度。这样做虽然不能提高频率分辨率，但补零后对原来的 $X(k)$起到了插值作用，可以平滑频谱的包络。但是，如果增加有效数据的长度，则可以提高频率分辨率。

【例 4-8】设信号 $x(n)$=sin(3.1*2*pi*n/fs)+cos(3*2*pi*n/fs)。$x(n)$具有两个频率分量 f_1=3.1Hz，f_2=3Hz，取采样频率 f_s=20Hz。验证补零和增加有限数据长度对频率分辨率的影响。

由前面的关系式，可计算出分辨这两个频率 f_1 和 f_2 的 N 值，要求 N>400。其实现的 MATLAB 程序代码如下：

```
>> clear all;
% 计算 0≤n≤128 时,x(n)的 DFT
```

```
N=128;                                          %数据长度
n=0:N-1;
fs=20;                                          %采样率
xn=sin(3.1*2*pi*n/fs)+cos(3*2*pi*n/fs);         %计算 x(n)
Xk=DFT(xn);                                     %x(n)的 DFT
figure;
subplot(2,1,1);plot(n,xn);
xlabel('(a) x(n)    (0≤n≤128) ');
m=(0:N/2-1)*fs/N;
subplot(2,1,2);plot(m,abs(Xk(1:N/2)));
xlabel('(b) x(n)进行 DFT 的结果');
% 在前面的 x(n)的后面补 128 个零,计算 DFT
N1=256;                                         %数据长度
n1=0:N1-1;
xn1=[xn,zeros(1,N1-N)];                         %将 x(n)补零
Xk1=DFT(xn1);
figure;
subplot(2,1,1);plot(n1,xn1);
xlabel('(a) x(n)    (补零后)');
m1=(0:N1/2-1)*fs/N1;
subplot(2,1,2);plot(m1,abs(Xk1(1:N1/2)));
xlabel('(b)  补零后的 DFT 结果');
% 计算 0≤n≤401 时,x(n)的 DFT
N2=401;                 %数据长度
n2=0:N2-1;
xn2=sin(3.1*2*pi*n2/fs)+cos(3*2*pi*n2/fs);    %计算 x(n)
Xk2=DFT(xn2);
figure;
subplot(2,1,1);plot(n2,xn2);
xlabel('(a) x(n)    (0≤n≤401)');
m2=(0:N2/2-1)*fs/N2;
subplot(2,1,2);plot(m2,abs(Xk2(1:N2/2)));
xlabel('(b) x(n)进行 DFT 的结果');
```

运行程序，效果如图 4-8～图 4-10 所示。

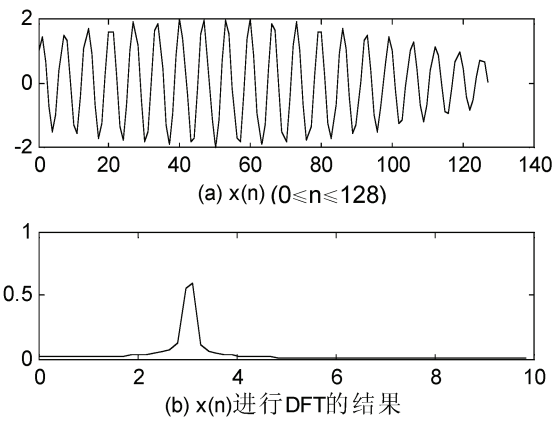


图 4-8　0≤n≤128 时 x(n)进行 DFT 的结果

经过对比可知，显然，图 4-9 所示的 DFT 结果的包络明显比图 4-8 所示的 DFT 包络平滑，但是都无法分辨出两个频率分量。而根据公式计算得出。当 N>400 时，能分辨出 $x(n)$中的两个频率分量。从图 4-10 可以看出，增加有效数据长度，可以提高分辨率，该结论完全正确。

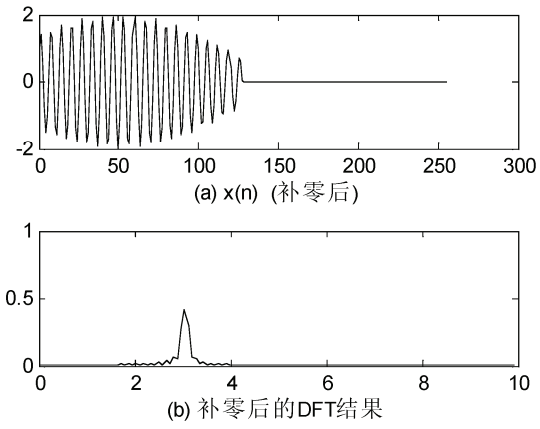


图 4-9　将 x(n)进行补零后，作 DFT 的结果

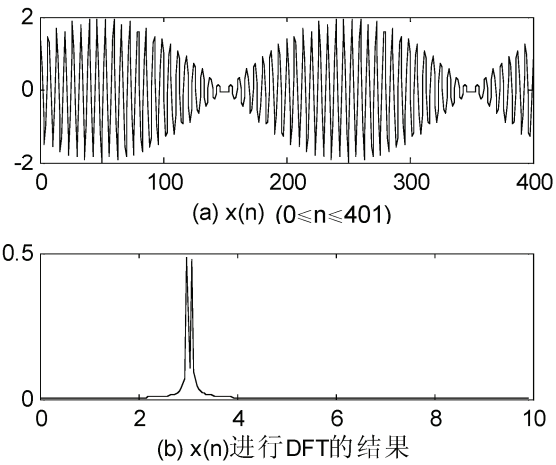


图 4-10　增加 x(n)数据有效长度 0≤n≤401，进行 DFT 的结果

4.3　快速傅里叶变换

4.3.1　快速傅里叶变换（FFT）的性质

1．按时间抽取的基-2FFT 算法

1）基本原理

DFT 计算公式为

$$\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{nk}\quad(k=0,1,\cdots,N-1)\tag{4-42}$$

设输入序列长度为 $N=2^L$（L 为正整数），将序列按时间顺序的奇偶分解为越来越短的子序列，称为按时间抽取的 FFT 算法，也称 Cooley-Tukey 算法。

步骤：

（1）先将 $x(n)$按 n 的奇偶分为两组作变量置换，即当 n 为偶数时，令 $n=2r$，当 n 为奇数时，令 $n=2r+1$。这样变量置换后，得

$$x(2r)=x_1(r),\quad x(2r+1)=x_2(r)\quad \left(r=0,1,2,\cdots,\frac{N}{2}-1\right) \tag{4-43}$$

则其 DFT 可化为两个部分：

$$\begin{aligned}X(k)&=\sum_{n=0}^{N-1}x(n)W_N^{nk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_N^{2rk}+\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_N^{(2r+1)k}\\&=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}+W_N^{nk}\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{\frac{N}{2}}^{rk}\\&=X_1(k)+W_N^kX_2(k)\quad \left(k=0,1,\cdots,\frac{N}{2}-1\right)\end{aligned} \tag{4-44}$$

这里用到了性质：

$$W_N^{2nk}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}2nk}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{\frac{N}{2}}nk}=W_{\frac{N}{2}}^{nk} \tag{4-45}$$

式（4-44）中，$X_1(k)$和 $X_2(k)$都为 $\frac{N}{2}$ 点的 DFT，即

$$X_1(k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{rk}\quad \left(k=0,1,\cdots,\frac{N}{2}-1\right) \tag{4-46}$$

$$X_2(k)=\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{\frac{N}{2}}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_{\frac{N}{2}}^{rk}\quad \left(k=0,1,\cdots,\frac{N}{2}-1\right) \tag{4-47}$$

由于序列 $x_1(r)$和 $x_2(r)$的长度为 $\frac{N}{2}$，$X_1(k)$和 $X_2(k)$的长度也是 $\frac{N}{2}$，而 $X(k)$却有 N 个点，所以求 $X(k)$可分为前 $\frac{N}{2}$ 点和后 $\frac{N}{2}$ 点。

前半部分：

$$X(k)=X_1(k)+W_N^kX_2(k)\quad \left(k=0,1,\cdots,\frac{N}{2}-1\right)$$

后半部分：

$$\begin{aligned}X\left(\frac{N}{2}+k\right)&=X_1\left(\frac{N}{2}+k\right)+W_N^{\left(\frac{N}{2}+k\right)}X_2\left(\frac{N}{2}+k\right)\\&=X_1(k)-W_N^kX_2(k)\quad \left(k=0,1,\cdots,\frac{N}{2}-1\right)\end{aligned}$$

这里用到了 W_N^k 的对称性质：$W_N^{\left(\frac{N}{2}+k\right)}=W_N^{\frac{N}{2}}W_N^k=-W_N^k$。

只要求出 $k=0,1,\cdots,\frac{N}{2}-1$ 区间内各点的 $X_1(k)$和 $X_2(k)$，即可求出 k=0,1,⋯,N−1 区间全部 $X(k)$值，这正是 FFT 能大量节省计算量的关键所在。

（2）由于 $N=2^L$，因此 $N/2$ 仍是偶数，可以进一步把 $N/2$ 点子序列 $x_1(r)$和 $x_2(r)$，再按 r 偶分解为两个 $N/4$ 点的子序列。按照这种方法不断划分下去，直到最后剩下的是两点 DFT 为止。两点的 DFT 实际上只有加减运算。

2）性质

（1）同址运算。从流图或蝶形结中可以看出，某一列的任何两个节点 k 和 j 的节点变量进行蝶形运算后，得到的结果为下一列 k 和 j 两节点的节点变量，而与其他节点变量无关，因而可以采用同址运算，即某一列 N 个数据送到存储器后，经蝶形运算，其结果为另一列数据，它们以蝶形为单位仍存储在同一组存储器中，直到最后输出，中间无需其他存储器。这样整个 FFT 运算只需 N 个存储单元。

（2）运算量。$N=2^L$ 的基-2FFT 算法共有 L 级蝶形，每级都由 $N/2$ 个蝶形结运算组成，每个蝶形结需要一次复乘、二次复加，因而每级运算都需 $N/2$ 次复乘和 N 次复加，这样 L 级运算总共需要：

复乘次数：

$$m_F=\frac{N}{2}L=\frac{N}{2}\log_2 N \tag{4-48}$$

复加次数：

$$\alpha_F=NL=N\log_2 N \tag{4-49}$$

FFT 算法与直接 DFT 算法所需的复乘运算量相比为

$$\frac{N^2}{\frac{N}{2}\log_2 N}=\frac{2N}{\log_2 N} \tag{4-50}$$

（3）倒位序的问题。

① 造成倒位序的原因在于输入 $x(n)$按标号 n 的偶奇不断分组。

② 倒位序实际上是将序号 n 表示成 L 位（$N=2^L$）二进制数，将此二进制数位序颠倒，即可得到对应的倒位序数。

（4）按时间抽取的其他形式流图。只要保持各节点所连接的支路及其传输系数不变，则不论节点位置怎么排，所得流图总是等效的，最后结果都是 DFT 的正确结果，只是数据的存放不同而已，这样就可得到按时间抽取的 FFT 算法的其他形式流图。

3）结论

FFT 算法并不是一种新的变换，而只是 DFT 的快速算法，因而 DFT 的理论、性质和约束条件在 FFT 中都是适用的。

在基-2FFT 算法中，如果序列长度 N 不满足 2 的整数倍的条件，则需要人为地在序列尾加上零值点，直到序列的长度达到要求，这样添加零值点时域序列值并没有变化，不影响其频率响应，只是使频率抽样间隔变密。

2．按频率抽取的基-2 FFT 算法

1）基本原理

$$\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{nk}\quad(k=0,1,\cdots,N-1)\tag{4-51}$$

设序列 $x(n)$长度为 $N=2^L$（L 为正整数），把它的 DFT$X(k)$（也是 N 点序列）按其序号的奇偶分解为越来越短的子序列，这种 FFT 算法称为按频率抽取的 FFT 算法。

2）步骤

（1）在将输出 $X(k)$按 k 的偶奇分组前，先把输入 $x(n)$按 n 的顺序分为前后两半：

$$\begin{aligned}X(k)&=\sum_{n=0}^{N-1}x(n)W_N^{nk}=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{nk}+\sum_{n=\frac{N}{2}}^{N-1}x(n)W_N^{nk}\\&=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{nk}+\sum_{n=\frac{N}{2}}^{\frac{N}{2}-1}x\left(n+\frac{N}{2}\right)W_N^{(n+\frac{N}{2})k}\\&=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{nk}+W_N^{(\frac{N}{2})k}\sum_{n=\frac{N}{2}}^{\frac{N}{2}-1}x\left(n+\frac{N}{2}\right)W_N^{nk}\quad(k=0,1,\cdots,N-1)\end{aligned}\tag{4-52}$$

由于 $W_N^{\left(\frac{N}{2}\right)k}=(-1)^k$，所以，又可得

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{nk}+(-1)^k\sum_{n=\frac{N}{2}}^{\frac{N}{2}-1}x\left(n+\frac{N}{2}\right)W_N^{nk}\quad(k=0,1,\cdots,N-1)\tag{4-53}$$

（2）按 k 的奇偶将 $X(k)$分成两部分，再按频率 k 的奇偶将一个 N 点 DFT 分解为两个 $N/2$ 点 DFT。

令

$$\begin{cases}k=2r\\k=2r+1\end{cases}\left(r=0,1,2,\cdots,\frac{N}{2}-1\right)\tag{4-54}$$

则有

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+x\left(n+\frac{N}{2}\right)]W_N^{n(2r)}=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+x\left(n+\frac{N}{2}\right)]W_{\frac{N}{2}}^{nr}\tag{4-55}$$

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)-x\left(n+\frac{N}{2}\right)]W_N^{n(2r+1)}=\sum_{n=0}^{\frac{N}{2}-1}[x(n)-x\left(n+\frac{N}{2}\right)]W_N^nW_{\frac{N}{2}}^{nr}\tag{4-56}$$

令

$$\begin{cases}x_1(n)=x(n)+x\left(n+\frac{N}{2}\right)\\x_2(n)=[x(n)-x\left(n+\frac{N}{2}\right)]W_N^n\end{cases}\left(n=0,1,2,\cdots,\frac{N}{2}-1\right)\tag{4-57}$$

则有

$$\begin{cases} X(2r)=\sum_{n=0}^{\frac{N}{2}-1}x_1(n)W_{\frac{N}{2}}^{nr} \\ X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}x_2(n)W_{\frac{N}{2}}^{nr} \end{cases} \quad \left(r=0,1,2,\cdots,\frac{N}{2}-1\right) \tag{4-58}$$

这样就按频率 k 的奇偶将一个 N 点 DFT 分解为两个新序列 $x_1(n)$和 $x_2(n)$的 $N/2$ 点 DFT。

（3）由于 $N=2^L$，故 $N/2$ 仍是偶数，因而可将每个 $N/2$ 点 DFT 的再分解，一直进行到第 L 步，变成求 $N/2$ 个 2 点的 DFT 为止。

3）结论

（1）同址运算：与时间抽取法一样，由蝶形运算构成。

（2）DIF 蝶形运算结构。蝶形运算共有 L 级（$L=\log_2 N$），每级 $N/2$ 个蝶形结。

（3）DIF 与 DIT 蝶形结的关系。

① 互为转置。转置将流图的所有支路方向都反向，并且交换输入与输出，但节点变量值不变。

② 运算量相同。DIF 基本蝶形结与 DIT 的基本蝶形结复乘出现的位置不同，但运算量相同。

③ DIF 与 DIT 的根本区别不在于输入与输出节点的奇偶，而在于蝶形结。运算量：整个 FFT 运算所需的运算量与时间抽取法相同。

基-2FFT 算法在实际中使用得最多，因为它的程序简单、效率高、使用方便。如果 N 不满足 $N=2^L$，则可以将 $x(n)$补零使 N 增长到最近的 2^L 数值。但是如果要求准确的 N 点 DFT，而 N 又是一个复合数，则可以用混合基 FFT 算法。

3．基-4FFT 算法

定义：在 $N=r_1r_2,\cdots,r_L$ 且 $r_1=r_2=\cdots=r_L=4$ 的特殊情况下，即当 $N=4^L$ 时，混合基算法变成基-4FFT 算法。

举例：以 $N=16=4^2$ 为例，即 $L=2$；$n=4n_1+n_0$，$k=4k_1+k_0$，则有

$$X(k)=\sum_{n_0=0}^{3}\sum_{n_1=0}^{3}x(n_1,n_0)W_N^{4n_1k_0}W_N^{n_0k_0}W_N^{4n_0k_1} \tag{4-59}$$

因此，有

$$X_1(k_0,n_0)=\sum_{n_1=0}^{3}x(n_1,n_0)W_N^{n_1k_0} \tag{4-60}$$

$$X_2(k_0,k_1)=\sum_{n_1=0}^{3}[X_1(k_0,n_0)W_N^{n_0k_0}]W_4^{n_0k_0} \tag{4-61}$$

整序后可得 $X_2(k_0,k_1)=X(k_0,k_1)$。

所以，它的基本运算有三步：

第一步，做 $X(n)$的 4 点 DFT（变量为 n_1,k_0），得到 $X_1(k_0,n_0)$。

第二步，将 $X_1(k_0,n_0)$乘旋转因子 $W_N^{n_0k_0}$ 后做 4 点的 DFT（变量为 n_0,k_1）得到 $X_2(k_0,k_1)$。

第三步，由于 $X_2(k_0,k_1)$的变量是 k_0 在前，k_1 在后，是基-4FFT 倒位序的序列，因此，将其变量整序后得到正常顺序输出的序列 $X(k_0,k_1)$。

说明：

① 基-4FFT 基于运算单元为 4 点 DFT。

② 除 4 点 DFT 外还要乘旋转因子，第一级不乘（相当于乘 $W_N^0=1$）。

③ 基-4FFT 运算的倒位序关系。

设 $n=(n_1n_0)_{4\times4}$，$n=4n_1+n_0$ 倒位序后 $\overline{n}=(n_0n_1)_{4\times4}$，$\overline{n}=4n_0+n_1$。

过程如下所示：

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
(n_1n_0)	00	01	02	03	10	11	12	13	20	21	22	23	30	31	32
n_0n_1	00	10	20	30	01	11	21	31	02	12	22	32	03	13	23
$\overline{n}$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11

4．快速傅里叶反变换

正变换：

$$\mathrm{DFT}[x(n)]=X(k)=\sum_{n=0}^{N-1}x(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}nk}=\sum_{n=0}^{N-1}x(n)W_N^{nk}\quad(k=0,1,\cdots,N-1)\tag{4-62}$$

反变换：

$$\mathrm{IDFT}[X(k)]=x(n)=\frac{1}{N}\sum_{n=0}^{N-1}X(k)\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}nk}=\frac{1}{N}\sum_{n=0}^{N-1}X(k)W_N^{-kn}\quad(n=0,1,\cdots,N-1)\tag{4-63}$$

比较 DFT 与 IDFT 的公式，可以看出只要把 DFT 运算中的每一个系数 W_N^{nk} 换成 W_N^{-kn}，并且最后再乘以常数 1/*N*，则所有时间抽取或频率抽取的算法都可以拿来计算 IDFT。

IDFT 具体的计算方法可分为两种：

1）改变 FFT 流图系数的方法

将 DFT 运算中的每个系数 W_N^{nk} 换成 W_N^{-kn} 再乘以常数 1/*N*（由于 $1/N=(1/2)^L$，由于 FFT 的流图共有 *L* 列蝶列，相当于每列都乘一个因子 1/2，这样 FFT 算法就可以用来计算 IDFT）。

2）直接利用 FFT 流图的方法

由于

$$x^*(n)=\frac{1}{N}\sum_{n=0}^{N-1}X(k)W_N^{nk}\tag{4-64}$$

因此，有

$$x(n)=\frac{1}{N}\left[\sum_{n=0}^{N-1}X^*(k)W_N^{nk}\right]^*=\frac{1}{N}\{\mathrm{DFT}[X^*(k)]\}^*\tag{4-65}$$

所以，作 IDFT 可分为三步：

（1）先取 *X*(*k*)的共轭，得出 *X**(*k*)。

（2）作 *X**(*k*)的 FFT，即得 *x**(*n*)。

（3）取 *x**(*n*)的共轭，并除以 *N*，即得 *x*(*n*)。

5．重叠保留法与重叠相加法

如果一个长序列 *x*(*n*)和短序列 *h*(*n*)进行线性卷积，若用 FFT 计算圆周卷积代替线性卷积，则需要把 *x*(*n*)和 *h*(*n*)分别补上很多零值点，如长序列 *x*(*n*)为 50 点，短序列 *h*(*n*)为 7 点，则要把它们都补零至 64>50+7−1 点，再进行圆周卷积，这样造成很大的浪费，可以采用重叠相加法或重叠保留法来处理。

1）重叠相加法

（1）将 $x(n)$分解为很多段，每段长为 L 点，L 与 M 数量级相同，用 $x_i(n)$表示 $x(n)$的第 i 段：

$$h(n)=\begin{cases}h(n), & 0\leqslant n\leqslant M-1\\ 0, & M\leqslant n\leqslant N-1\end{cases} \tag{4-66}$$

$$x(n)=\sum_{i=0}^{\infty}x_i(n)$$

（2）再分别将 $x(n)$和 $h(n)$补零值点，补到长度为 N 点，N 满足：

$$N=2^m\geqslant L+M-1$$

（3）用 FFT 实现 N 点的圆周卷积。根据圆周卷积定理，用圆周卷积 $y_i=x_i(n)\Theta h(n)$代替线性卷积 $y_i=x_i(n)\otimes h(n)$，$x_i(n)$和 $h(n)$需分别补零到 N 点：

$$x_i(n)=\begin{cases}x(n), & iL\leqslant n\leqslant (i+1)L-1\\ 0, & (i+1)L\leqslant n\leqslant (i+1)L+N-1\end{cases}\quad (i=0,1,2,\cdots) \tag{4-67}$$

$$h(n)=\begin{cases}h(n), & 0\leqslant n\leqslant M-1\\ 0, & M\leqslant n\leqslant N-1\end{cases} \tag{4-68}$$

计算 N 点 FFT：

$$X_i(k)=\mathrm{DFT}[x_i(n)] \tag{4-69}$$

相乘：

$$Y_i(k)=H(k)X_i(k) \tag{4-70}$$

计算 N 点 IFFT：

$$y_i(k)=\mathrm{IDFT}[Y_i(n)] \tag{4-71}$$

将各段 $y_i(k)$相加：

$$y(n)=\sum_{i=0}^{\infty}y_i(n) \tag{4-72}$$

输出序列 $y(n)$：

$$y(n)=\sum_{i=0}^{\infty}x_i(n)*h(n)=\sum_{i=0}^{\infty}y_i(n) \tag{4-73}$$

由于 $x_i(n)$长度为 L 点，而 $y_i(k)$长度为 $L+M-1$ 点，则相邻两段输出序列有$(M-1)$点的重叠，将重叠部分直接相加。因此称此方法为重叠相加法。

用户自定义编写的重叠相加法计算卷积的 MATLAB 源程序如下：

```
function [y]=ovrlpsav(x,h,N)
% 用混叠相加法作块卷积
% [y]=ovrlpsav(x,h,N)
% y 为输出序列
% x 为输入序列
% h 为脉冲响应
% N 为块长
%
Len=length(x);
M=length(h);
M1=M-1;L=N-M1;
h=[h zeros(1,N-M)];
x=[zeros(1,M1),x,zeros(1,N-1)];      %预置(M-1)个零
```

```
K=floor((Len+M1-1)/(L));              %块数
Y=zeros(K+1,N);
% 与后续各块卷积
for i=0:K
    xi=x(i*L+1:i*L+N);
    Y(i+1,:)=dirconvt(xi,h,N);
end
Y=Y(:,M:N)';                          %去掉前(M-1)个样本
y=(Y(:))';                            %转置输出
```

而在程序中调用的用户自定义编写 dirconvt 函数实现序列的圆周卷积的 MATLAB 源代码如下：

```
function y=dirconvt(x1,x2,N)
% 在 x1 和 x2:(时域)之间的 N 点圆周卷积
% y=dirconvt(x1,x2,N)
% y 为包含圆周卷积的输出序列
% x1 为长度,N1<=N 的输入序列
% x2 为长度,N2<=N 的输入序列
% N 为圆周缓冲器的大小
% 方法 y(n)=sum(x1(m)*x2(n-m) mod N)
% 检查 x1 的长度
if length(x1)>N
    error('N 必须>=x1 的长度');
end
% 检查 x2 的长度
if length(x2)>N
    error('N 必须>=x2 的长度');
end
x1=[x1 zeros(1,N-length(x1))];
x2=[x2 zeros(1,N-length(x2))];
m=[0:1:N-1];
x2=x2(mod(-m,N)+1);
H=zeros(N,N);
for n=1:1:N
    H(n,:)=cirshftt(x2,n-1,N);
end
y=x1*H';;
% 计算 5 点、6 点循环卷积可以实现如下
% 5 点循环卷积
x1=[1,2,2];x2=[1,2,3,4];
y=dirconvt(x1,x2,5)
% 6 点循环卷积
x1=[1,2,2];x2=[1,2,3,4];
y=dirconvt(x1,x2,6)
```

2）重叠保留法

用 FFT 算法实现重叠保留法的步骤为：先将 $x(n)$分段，每段长度为 $L=N-M+1$ 个点（M 为短序列长度，N 为大于短序列长度的 2 的整次幂数，如 $M=7$，$N=16$）。分段后在每一段的前边补上前一段保留下来的$(M-1)$个输入序列值，组成 N 点短序列。对于第一段，由于没有前一段

保留信号，则需要在它前边填充 M–1 个零值点。

（1）计算 N 点 FFT：

$$H_j(k)=\mathrm{DFT}[h_j(k)] \tag{4-74}$$

（2）计算 N 点 FFT：

$$X_j(k)=\mathrm{DFT}[x_j(k)] \tag{4-75}$$

（3）相乘：

$$Y_j(k)=H(k)X_j(k) \tag{4-76}$$

（4）计算 N 点 IFFT：

$$y_j(k)=\mathrm{IDFT}[Y_j(n)] \tag{4-77}$$

（5）输出序列 $y(n)$：

$$y(n)=\sum_{j=0}^{\infty}x_j(n)*h(n)=\sum_{i=0}^{\infty}y_j(n) \tag{4-78}$$

每段圆周卷积结果的前(M–1)个点不等于线性卷积值，必须舍去。再把各相邻输出段留下来的序列衔接起来，就构成了最后的正确输出。

与重叠相加法的比较：

（1）两者分段的方法相同，每段 $L=N–M+1$ 个点。

（2）两者补点的方法不同。在补齐 N 点时，重叠保留法则在每一分段序列前边补上一段保留下来的(M–1)个输入序列值（第一分段前补(M–1)个零点），重叠相加法是用在每一分段序列后补零点的方法。

（3）两者由每段结果组成最后输出序列的方法不同。重叠保留法是将各相邻段重叠部分舍去后衔接起来构成最后输出；重叠相加法是将各相邻段重叠部分相加再和不重叠的部分共同组成最后输出。

采用 FFT 高速分段卷积的重叠保留法的 MATLAB 程序如下：

```
function [y]=hsolpsav(x,h,N)
% 用 FFT 的高速混叠法作循环卷积
% [y]=hsolpsav(x,h,N)
% y 为输出序列
% x 为输入序列
% h 为脉冲响应
% N 为块长（必须是 2 的幂次）
%
N=2^(ceil(log10(N)/log10(2)));
Len=length;
M=length(h);
M1=M-1;L=N-M1;
h=fft(h,N);
x=[zeros(1,M1),x,zeros(1,N-1)];
K=floor((Len+M1-1)/(L));   %块的数目
Y=zeros(K+1,N);
for i=0:K
    xi=fft(x(i*L+1:i*L+N));
    Y(i+1,:)=real(ifft(xi.*h));
end
```

```
Y=Y(:,M:N)';
y=(Y(:))';
```

4.3.2 快速傅里叶变换及其应用

前面已经介绍过傅里叶变换，对其用法及各种性质进行了介绍。快速傅里叶变换与离散傅里叶变换的理论分析完全一致，只不过运算速度加快而已。

在 MATLAB 信号处理工具箱中函数 FFT 和 IFFT 用于快速傅里叶变换和逆变换。快速傅里叶变换函数调用格式如下：

y = fft(x)

式中：x 是序列；y 是序列的快速傅里叶变换。x 可以为一向量或矩阵，若 x 为向量，则 y 是 x 的的 FFT，并且与 x 具有相同的长度。若 x 为一矩阵，则 y 是对矩阵的每一列向量进行 FFT。

如果 x 的长度为 2 的整数次幂，函数 FFT 执行高速其-2FFT 算法；否则 FFT 执行一种混合基的离散傅里叶算法，计算速度较慢。这就是说，只有当 x 的长度为 2 的整数次幂才能最大限度地提高程序运算速度。

函数 FFT 的另一种调用形式如下：

y = fft(x,n)

式中：x, y 的意义同前；n 为正整数。此时函数执行 n 点的 FFT。若 x 为向量且长度小于 n，则函数将 x 补零至长度 n，若向量 x 的长度大于 n，则函数截断 x 使之长度为 n。

对应于快速傅里叶变换函数 FFT，MATLAB 信号处理工具箱中提供的快速傅里叶反变换函数如下：

y = ifft(X)和 y = ifft(X,n)

式中：X 为需要进行逆变换的序列信号，一般情况下为复数；y 为 IFFT 的输出，通常包含实部和虚部两部分。n 的意义与 FFT 中的一样。

【例 4-9】已知信号 x(t)=0.5*sin(2*pi*f1*t)+2*sin(2*pi*f2*t)，其中 f1=15Hz，f2=40Hz，采样频率为 100Hz，在下列情况下绘制其幅频图。分析所用数据长度不同时对傅里叶变换结果的影响。

（1）数据个数 N=32，FFT 所用的采样点数 NFFT=32；

（2）N=32；NFFT=128；

（3）N=136；NFFT=128；

（4）N=136；NFFT=512。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
fs=100;                                  %采样频率
Ndata=32;                                %数据长度
N=32;                                    %FFT 的数据长度
n=0:Ndata-1;
t=n/fs;                                  %数据对应的时间序列
x=0.5*sin(2*pi*15*t)+2*sin(2*pi*40*t);   %时间域信号
y=fft(x,N);                              %信号的傅里叶变换
mag=abs(y);                              %求取振幅
f=(0:N-1)*fs/N;                          %真实频率
subplot(2,2,1);plot(f(1:N/2),mag(1:N/2)*2/N);
```

```
xlabel('频率/Hz');ylabel('振幅');
title('Ndata=32; NFFT=32');
grid on;

Ndata=32;                                   %数据长度
N=128;                                      %FFT 采用的数据长度
n=0:Ndata-1;
t=n/fs;                                     %时间序列
x=0.5*sin(2*pi*15*t)+2*sin(2*pi*40*t);      %时间域信号
y=fft(x,N);
mag=abs(y);
f=(0:N-1)*fs/N;                             %真实频率
subplot(2,2,2);plot(f(1:N/2),mag(1:N/2)*2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata=32; NFFT=128');
grid on;

Ndata=136;                                  %数据长度
N=128;                                      %FFT 采用的数据长度
n=0:Ndata-1;
t=n/fs;                                     %时间序列
x=0.5*sin(2*pi*15*t)+2*sin(2*pi*40*t);      %时间域信号
y=fft(x,N);
mag=abs(y);
f=(0:N-1)*fs/N;                             %真实频率
subplot(2,2,3);plot(f(1:N/2),mag(1:N/2)*2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata=136; NFFT=128');
grid on;

Ndata=136;                                  %数据长度
N=512;                                      %FFT 采用的数据长度
n=0:Ndata-1;
t=n/fs;                                     %时间序列
x=0.5*sin(2*pi*15*t)+2*sin(2*pi*40*t);      %时间域信号
y=fft(x,N);
mag=abs(y);
f=(0:N-1)*fs/N;                             %真实频率
subplot(2,2,4);plot(f(1:N/2),mag(1:N/2)*2/N);
xlabel('频率/Hz');ylabel('振幅');
title('Ndata=136; NFFT=512');
grid on;
```

程序运行结果如图 4-11 所示。当数据个数和 FFT 采用的数据个数均为 32 时，频率的分辨率较低，但没有由于添加零而导致的其他频率成分。这里由于将振幅谱*2/N，因此就得到了绝对大小的振幅。当数据个数为 32，FFT 采用 128 时，FFT 程序将数据后补零足 128 个数据。由于在时间域内信号加零，致使振幅谱中出现很多其他成分，这是加零造成的。其振幅由于加了许多零而明显减小。当数据个数增加到 136，FFT 所用数据个数为 128，则 FFT 程序将数据截

断为 128 个数据。这时分辨率较高；当数据个数为 136，FFT 所用数据为 512 时，也需要在原始数据后补加（512−136）个零。虽然如此，但由于含有信号的数据个数足够多，其傅里叶变换振幅谱也基本不受影响。

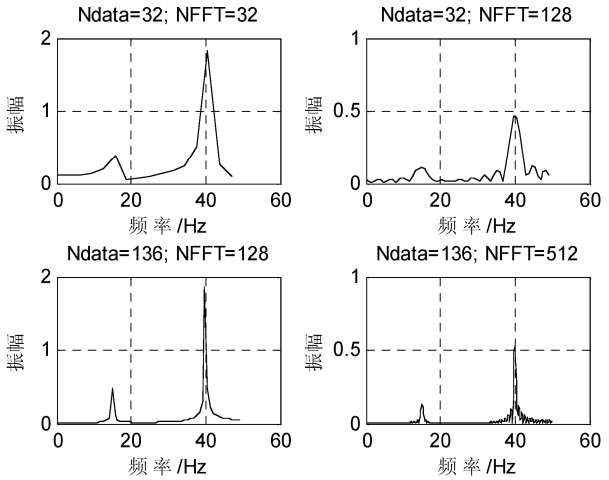


图 4-11　改变数据个数和 FFT 采用的数据个数对傅里叶谱的影响效果图

【例 4-10】对信号 x(t)=sin(2*pi*40*t)+sin(2*pi*15*t)进行 FFT，对其结果进行逆 FFT，将结果与原信号进行比较。采样频率为 100Hz，采样点数为 128。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
fs=100;                                  %采样频率
N=128;                                   %数据个数
n=0:N-1;
t=n/fs;                                  %数据对应的时间序列
x=0.5*sin(2*pi*15*t)+2*sin(2*pi*40*t);   %时间域信号
subplot(2,2,1);plot(t,x);
xlabel('时间/s');ylabel('x');
title('原始信号');
grid on;

y=fft(x,N);                              %傅里叶变换
mag=abs(y);                              %得到振幅谱
f=n*fs/N;                                %频率序列
subplot(2,2,2);plot(f(1:N/2),mag(1:N/2)*2/N);
xlabel('频率/Hz');ylabel('振幅');
title('原始信号的快速傅里叶变换');
grid on;

xifft=ifft(y);                           %进行傅里叶逆变换
realx=real(xifft);                       %求取傅里叶逆变换的实部
ti=[0:length(xifft)-1]/fs;               %傅里叶逆变换的时间序列
subplot(2,2,3);plot(ti,realx);
xlabel('时间/s');ylabel('x');
```

```
title('运用傅里叶逆变换得到的信号');
grid on;

yif=fft(xifft,N);                          %将傅里叶逆变换得到的时间域信号进行傅里叶变换
mag=abs(yif);
f=[0:length(y)-1]'*fs/length(y);           %频率序列
subplot(2,2,4);plot(f(1:N/2),mag(1:N/2)*2/N);
xlabel('频率/Hz');ylabel('振幅');
title('运用 IFFT 得到信号的快速傅里叶变换');
grid on;
```

运行程序结果如图 4-12 所示。可以看到对傅里叶变换后的频谱进行逆变换得到的时间域信号与原始时间域信号完全一致。程序中对其进行取实部运算，如果大家分析一下其虚部，可以发现基本为零（不为零是由于数据截断造成的；如果数据无限长，则可得到为零的结果）。得到的时间域信号再次进行傅里叶变换后与原始信号的傅里叶变换后结果一致。

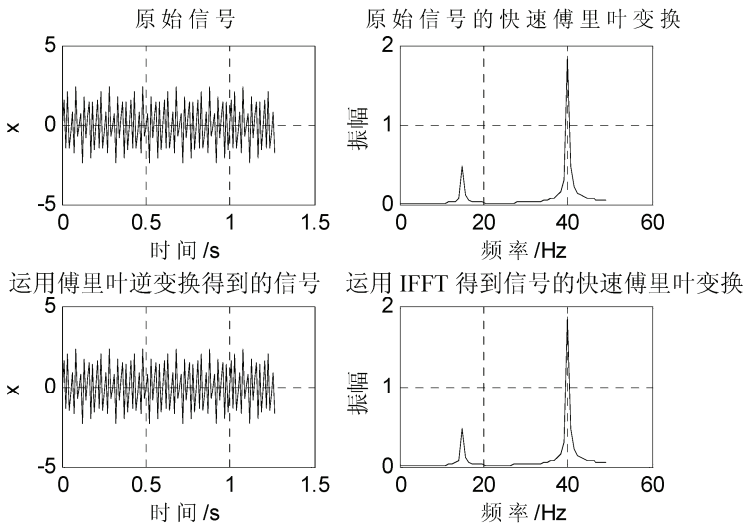


图 4-12　将信号进行傅里叶变换后的逆变换结果与原时间域信号的比较

4.3.3　运用快速傅里叶变换进行简单滤波

由前面的分析可知，根据 FFT 可以知道信号序列中含有哪些频率成分，各频率成分的振幅是多大。根据 IFFT，可以把频率域的信号转化回时间域，从而得到与原信号长度相同的时间序列。那么，能否可以通过将频率域中的某些频率成分振幅置零，然后运用傅里叶反变换到时间域而达到滤波的效果呢？回答是肯定的。这时一个自然提出的问题是，若将某些频率的振幅置零，其相位信息不变，这样会不会有问题？但可以想到，若该频率信号的振幅为零，其相位根本不起作用。注意：由于 FFT 得到的频率域一般只考虑奈奎斯特频率之前的频率，但当采用 FFT 滤波时，必须考虑奈奎斯特频率之后的振幅及相位。

【例 4-11】运用快速傅里叶变换对信号 x=0.5*sin(2*pi*3*n*dt)+cos(2*pi*10*n*dt)，数据点数为 512，进行滤波，将频率为 8Hz～15Hz 的波滤除掉。采样时间间隔 dt=0.02。绘出滤波前和滤波后的振幅谱以及滤波后的时间域信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
dt=0.02;N=512;
n=0:N-1; t=n*dt;                          %时间序列
f=n/(N*dt);                               %频率序列
f1=3; f2=10;                              %信号的频率成分
x=0.5*sin(2*pi*f1*t)+cos(2*pi*f2*t);
subplot(2,2,1);plot(t,x);                 %绘制原始的信号
title('原始信号的时间域');xlabel('时间/s');
y=fft(x);                                 %对原信号作 FFT 变换
xlim([0 12]);ylim([-1.5 1.5]);
subplot(2,2,2);plot(f,abs(y)*2/N);        %绘制原始信号的振幅谱
xlabel('频率/Hz');ylabel('振幅');
xlim([0 50]);title('原始振幅谱');
ylim([0 0.8]);
f1=8;f2=15;                               %要滤去频率的上限和下限
yy=zeros(1,length(y));                    %设置与 y 相同的元素数组
for m=0:N-1                               %将频率落在该频率范围及其大于奈奎斯特频率的波滤去
    % 小于 Nyquist 频率的滤波范围
    if (m/(N*dt)>f1 & m/(N*dt)<f2) | (m/(N*dt)>(1/dt-f2) & m/(N/dt)<(1/dt-f1));
    % 大于 Nyquist 频率的滤波范围
    % 1/dt 为一个频率周期
    yy(m+1)=0;                            %置在此频率范围内的振动振幅为零
    else
        yy(m+1)=y(m+1);                   %其余频率范围的振动振幅不变
    end
end
subplot(2,2,4);plot(f,abs(yy)*2/N)        %绘制滤波后的振幅谱
xlim([0 50]);ylim([0 0.5]);
xlabel('频率/Hz');ylabel('振幅');
gstext=sprintf('自%4.1f-%4.1fHz 的频率被滤除',f1,f2);
%将滤波范围显示作为标题
title(gstext);
subplot(2,2,3);plot(t,real(ifft(yy)));
%绘制滤波后的数据运用 ifft 变换回时间域并绘图
title('通过 IFFT 回到时间域');
xlabel('时间/s');
ylim([-0.6 0.6]);xlim([0 12]);
```

运行程序效果如图 4-13 所示。可见无论在时间域或频率域均滤除了 8Hz～15Hz 的频率成分（10Hz 的频率成分）。大家可以选择不同的滤波范围进行试验或设计其他的信号进行试验。时间域显示其滤波效果还是相当好的，这是最“彻底”、最“干净”的滤波。这种滤波的缺点是由于使用傅里叶变换，相比于后面所讲的滤波技术，这种技术运算相对较慢。

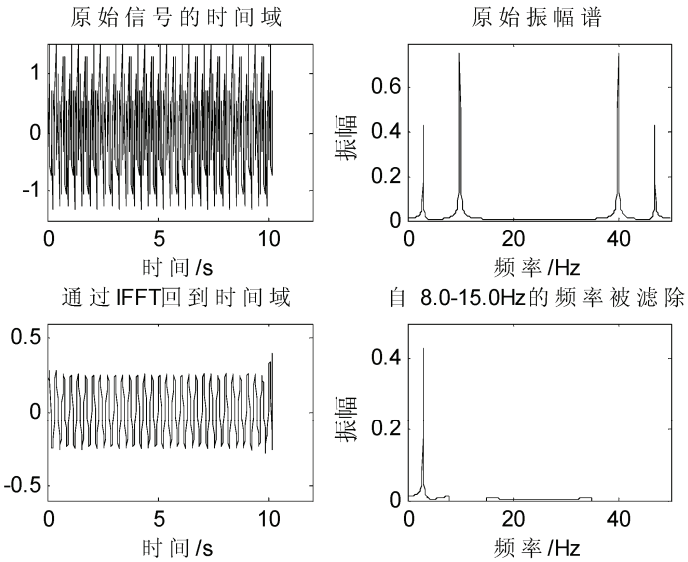


图 4-13　运用傅里叶变换滤除 8Hz～15Hz 频率成分效果图

4.4　离散余弦变换

与 DFT 相比，信号的离散余弦变换（DCT）具有更好的能量压缩性能，用少数几个变换系数就可以很好地表征信号的总体。正是 DCT 具有此性质，使得它在数据压缩和数据通信中得到广泛的应用，而且 DCT 避免了繁杂的复数运算。对于实信号的 DCT，其结果仍然是实信号。

长度为 N 的序列 $x(n)$，其 DCT 为

$$\begin{cases} X_c(0)=\dfrac{1}{\sqrt{N}}\displaystyle\sum_{n=0}^{N-1}x(n) \\ X_c(k)=\sqrt{\dfrac{2}{N}}\displaystyle\sum_{n=0}^{N-1}x(n)\cos\dfrac{(2n+1)k\pi}{2N} \quad (k=1,2,\cdots,N-1) \end{cases} \tag{4-79}$$

令

$$C_{k,n}=\sqrt{\frac{2}{N}}g_k\cos\frac{(2n+1)k\pi}{2N}$$

其中

$$g(k)=\begin{cases}\dfrac{1}{\sqrt{2}}, & k=0\\ 1, & k\neq 0\end{cases}$$

则序列 $x(n)$的 DCT 为

$$\boldsymbol{X}_c=\boldsymbol{C}_N\boldsymbol{x} \tag{4-80}$$

由于变换矩阵 C_N 为归一化的正交阵，所以，DCT 是正交变换。这样，如果序列 $x(n)$为实数序列，则其 DCT 也为实数。而对于傅里叶变换，即使序列 $x(n)$为实数，其 DFT 也一般为复数。因此 DCT 避免了复数运算。

离散余弦反变换（IDCT）定义为

$$x(n)=\frac{1}{\sqrt{N}}X_c(0)+\sqrt{\frac{2}{N}}X_c(k)\frac{(2n+1)k\pi}{2N},\quad n=0,1,2,\cdots,N-1 \tag{4-81}$$

同样，利用变换矩阵可以将 IDCT 写成如下形式：

$$\boldsymbol{x}=\boldsymbol{C}_N^{-1}\boldsymbol{X}_c=\boldsymbol{C}_N^{\mathrm{T}}\boldsymbol{X}_c \tag{4-82}$$

在 MATLAB 信号处理工具箱中，提供了用于 dct 和 idct 变换的内部函数，dct 函数其调用格式如下：

y = dct(x)：返回序列 x 的 DCT 结果。如果 x 为矩阵（多通道信号），则返回的是 x 中每一列信号的 DCT。

y = dct(x,n)：则将序列 x 补零到或截断到长度为 n 后，然后再进行 DCT。

【例 4-12】序列 x = (1:100) + 50*cos((1:100)*2*pi/40)，计算其 DCT。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x = (1:100) + 50*cos((1:100)*2*pi/40);
X = dct(x);
[XX,ind] = sort(abs(X)); ind = fliplr(ind);
i = 1;
while (norm([X(ind(1:i)) zeros(1,100-i)])/norm(X)<.99)
    i = i + 1;
end
i;
subplot(1,2,1);plot(x);
xlabel('(a)  原始信号');
subplot(1,2,2);plot(X)
xlabel('(b) DCT 效果');
```

运行程序，输出如下，效果如图 4-14 所示。

```
i =
     3
```

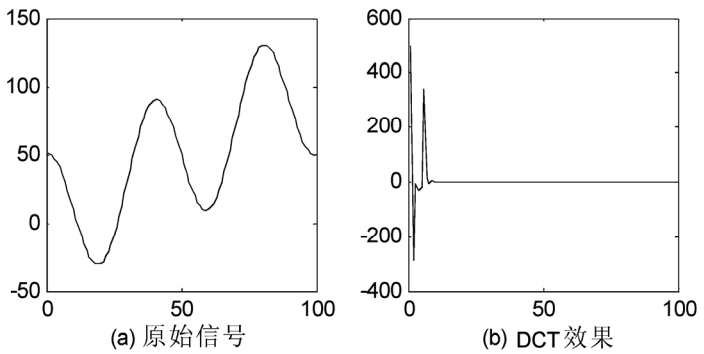


图 4-14　信号的 DCT

idct 函数的调用格式如下：

x = idct(y)：返回 x 的 idct 变换结果。若 y 为矩阵（多通道信号），则返回的是 y 中每一列信号的 idct 变换。

x = idct(y,n)：是将序列 y 补零或截断到长度 n 后，再进行 idct 变换。

【例 4-13】已知序列 x(n)为：x(n)=sin(2*pi*n*f/fs)，0≤n<200，其中 f=100Hz，fs=2000Hz，

计算该序列的 DCT，并用系数幅度大于 5 的部分来重建信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:200-1;
f=100;   fs=2000;
x=sin(2*pi*n*f/fs);
y=dct(x);                    %计算 DCT
m=find(abs(y<5));            %利用阈值对变换系数截取
y(m)=zeros(size(m));
z=idct(y);                   %对门限处理后的系数进行 IDCT
subplot(1,2,1);plot(n,x);
xlabel('n');title('x(n)');
subplot(1,2,2);plot(n,z);
xlabel('n');title('z(n)');
```

运行程序，效果如图 4-15 所示。

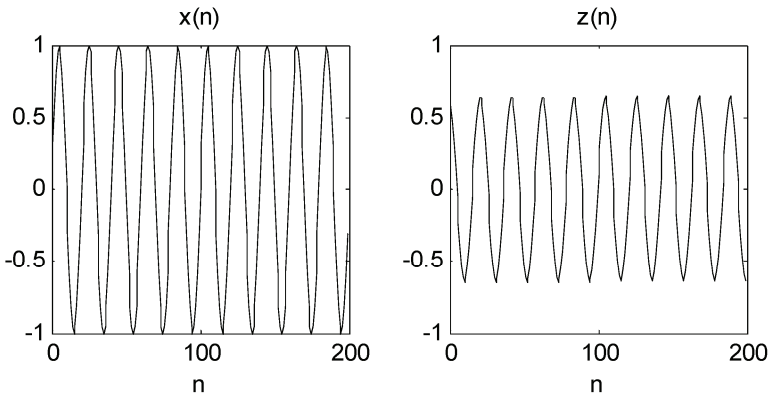


图 4-15　原始 x(n)和重建后的序列 z(n)

由图 4-15 可以看出，重建信号与原始信号差别不大，很好地恢复了原始信号。

4.5　Chirp Z 变换

DFT 可以看成是对信号在 Z 平面单位圆上均匀采样，但在实际应用中，并不是整个单位圆上的所有采样都是有意义的。比如，对于一个窄带信号，所需要分析的只是信号所在的一段频段。与 DFT 不同。Chirp Z 变换是一种更为灵活的计算频谱的方法，可以看成是 z 域上沿螺旋曲线的 Z 变换，可以用来计算单位圆上任意一段曲线上的 Z 变换。

作 DFT 时，输入信号的点数 N 和输出信号的点数 M 可以不相等，从而实现频域上的细化。

序列的 Z 变换为

$$X(z)=\sum_{n=0}^{\infty}x(n)z^{-n} \tag{4-83}$$

设 Z 变换的螺旋曲线为 $z_l=AW^{-1}$，其中下标 l 为螺旋任一条曲线标号，$l=0,1,\cdots,M-1$。

螺旋曲线的复数起点为 $A=A_0\mathrm{e}^{2\mathrm{j}\pi\theta_0}$，螺旋曲线上各点之间的比率为 $W=W_0\mathrm{e}^{-2\mathrm{j}\pi\varphi_0}$，$A_0$ 和 W_0 为任意的正实数。

螺旋曲线有以下特点：

（1）当 A_0>1 时，螺旋曲线在单位圆外，反之在单位圆内。

（2）当 W_0>1 时，A_0W^{-1}<A_0，螺旋曲线内旋，反之螺旋曲线外旋。

（3）当 A_0=W_0=1 时，Chirp Z 变换的变换路径为单位圆上的一段圆弧。

（4）当 A_0=W_0=1，θ_0=0，M=N 时，Chirp Z 变换为普通的 Z 变换。

沿螺旋曲线的 Z 变换（即 Chirp Z 变换）为

$$X(z_l)=\sum_{n=0}^{N-1}x(n)(AW^{-t})^{-n}=\sum_{n=0}^{N-1}x(n)A^{-n}W^{nl}z^{-n}\ ,\ 0\leqslant l\leqslant M-1 \qquad (4\text{-}84)$$

因此，如果希望得到信号的频谱分析，则应当在单位圆上实现 Chirp Z 变换，即应取 A_0=W_0=1。

MATLAB 信号处理工具箱提供了内部函数 czt 用于实现 Chirp Z 变换，其调用格式如下：

y = czt(x,m,w,a)：用于计算指定参数 M、W、A 下的 Chirp Z 变换。

y=czt(x)：则使用默认参数进行 Chirp Z 变换。默认参数为：m=length(x)，w=exp(j*2*pi/m)，a=1。

【例 4-14】利用 Chirp Z 变换分析某滤波器的频率特性。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',0);
x = randn(1013,1);
y = czt(x);
h = fir1(30,125/500,rectwin(31));   % 滤波器
%建立频率与 CZT 参数
fs = 1000; f1 = 100; f2 = 150;
m = 1024;
w = exp(-j*2*pi*(f2-f1)/(m*fs));
a = exp(j*2*pi*f1/fs);
%建立 DFT 和 CZT 两个滤波器
y = fft(h,1000);
z = czt(h,m,w,a);
%创建频率向量并比较其结果
fy = (0:length(y)-1)'*1000/length(y);
fz = ((0:length(z)-1)'*(f2-f1)/length(z)) + f1;
subplot(1,2,1);plot(fy(1:500),abs(y(1:500)));
axis([1 500 0 1.2])
xlabel('(a) FFT')
subplot(1,2,2);plot(fz,abs(z),'r:');
axis([f1 f2 0 1.2])
xlabel('(b) Chirp Z 变换')
```

运行程序，效果如图 4-16 所示。

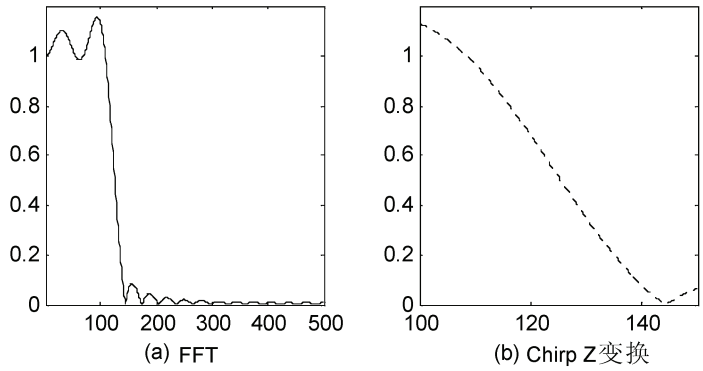


图 4-16　利用 Chirp Z 变换计算滤波器的频率响应特性

4.6　离散希尔伯特变换

在几乎所有利用傅里叶方法进行表示和分析物理过程的领域，傅里叶变换的实部和虚部之间或者幅度和相位之间，都存在一定关系，这就是希尔伯特（Hilbert）变换。希尔伯特变换是信号分析中重要工具，利用希尔伯特变换可以构造“解析信号”，使其仅含有正频分量，从而可以降低信号的采样率。

希尔伯特变换器的单位冲击响应 $h(n)$为

$$h(n)=\frac{1-(-1)^n}{n\pi}=\begin{cases}0, & n\text{ 为偶数}\\ \dfrac{2}{m}, & n\text{ 为奇数}\end{cases} \tag{4-85}$$

希尔伯特变换器的频域特性为

$$H(\mathrm{e}^{\mathrm{i}\omega})=\begin{cases}-\mathrm{j}, & 0<\omega<\pi\\ \mathrm{j}, & -\pi<\omega<\pi\end{cases} \tag{4-86}$$

则可以定义序列 $x(n)$的希尔伯特变换为

$$\hat{x}(n)=x(n)*h(n)=\frac{2}{\pi}\sum_{m=-\infty}^{\infty}\frac{x(n-2m-1)}{2m+1} \tag{4-87}$$

这样，可以构造解析信号 $z(n)$为

$$z(n)=x(n)+\mathrm{j}\hat{x}(n) \tag{4-88}$$

通过分析，可以证明解析信号 $z(n)$只含有正频分量，且是原信号的正频分量的 2 倍。

MATLAB 信号处理工具箱中，提供了计算希尔伯特变换的内部函数 hilbert，其调用格式如下：

x = hilbert(xr)：返回 x 的解析信号。y 的实部为原始信号 xr，虚部为 xr 的离散希尔伯特变换。

x = hilbert(xr,n)：采用 N 点 FFT 计算希尔伯特变换。

【例 4-15】计算正弦信号的希尔伯特变换。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=(0:1/1023:1);
x=sin(2*pi*60*t);
```

```
y=hilbert(x);
plot(t(1:50),real(y(1:50)));
hold on;
plot(t(1:50),imag(y(1:50)),':');
hold off;
legend('正弦信号的希尔伯特变换实部','正弦信号的希尔伯特变换虚部');
```

运行程序，效果如图 4-17 所示。

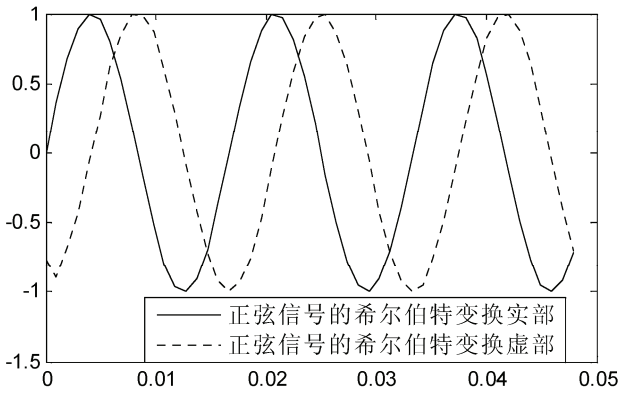


图 4-17　正弦信号的希尔伯特变换的实部和虚部

当然，除了利用 MATLAB 的内部函数 hilbert 来实现希尔伯特变换外，也可以用 DFT 求出信号 $x(n)$的解析信号及其希尔伯特变换。

首先，求 $x(n)$的 DFT $X(k)$，k=0,1,2,⋯,N−1，其中 k=N/2,⋯,N−1 对应于信号的负数频率。

$$z(k)=\begin{cases}X(k), & k=0\\ 2X(k), & k=1,2,\cdots,\dfrac{N}{2}-1\\ 0, & k=N/2,\cdots,N-1\end{cases} \tag{4-89}$$

再对 $z(k)$作 IDFT，即可得到 $x(n)$的解析信号。

最后由 $z(n)=X(k)+\mathrm{j}\hat{X}(k)$，得 $\hat{x}(n)=\mathrm{Im}[z(n)-x(n)]$。

利用 DFT 和希尔伯特变换理论，可以构造一个扩展的希尔伯特变换函数 dft_hilbert.m。其源代码如下：

```
function [z,hxm]=dft_hilbert(x)
N=length(x);
Xk=fft(x);
k1=0;
z(k1+1)=Xk(k1+1);
k2=1:N/2-1;
z(k2+1)=2*Xk(k2+1);
k3=N/2:N-1;
z(k3+1)=zeros(size(k3));
zn=ifft(z);
hxn=imag(zn-x);
```

利用上面的 dft_hilbert 扩展程序，对例 4-15 重新计算。其程序代码如下：

```
>> clear all;
```

```
t=(0:1/1023:1);
x=sin(2*pi*60*t);
[z,hxm]=dft_hilbert(x);
plot(t(1:50),x(1:50));
hold on;
plot(t(1:50),hxm(1:50),':');
hold off;
legend('正弦信号的希尔伯特变换实部','正弦信号的希尔伯特变换虚部');
```

运行程序，效果如图 4-18 所示。

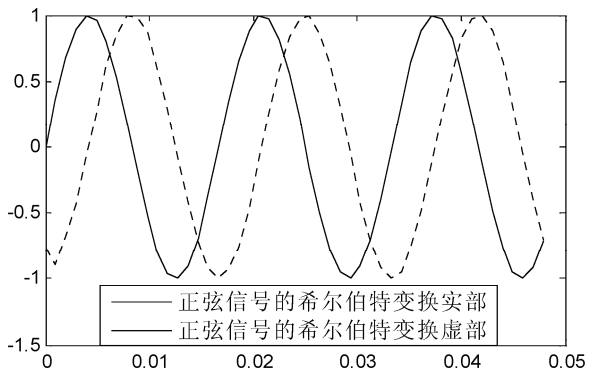


图 4-18　采用 dft_hilbert 函数计算 x(n)希尔伯特变换的实部和虚部

第 5 章　模拟滤波器

Wagner 和 Campbell 于 1915 年首次提出无源滤波器概念，从此滤波器开始了自身的蓬勃发展。滤波器发展的最大动力来源于微电子工业的发展，而现代计算机、DSP、大规模可编程逻辑器件、高速 D/A 的出现，加快了滤波器的数字化进程。

5.1　模拟滤波器的基本概念

滤波是信号处理的一种基本而重要的技术，利用滤波可从复杂的信号中提取所需要的信号，抑制不需要的部分。所谓滤波器，就是对已知激励，可以在时间域或频域产生规定响应的网络。要使滤波器能够提取有用信号，要求滤波器对信号与噪声有不同的增益，对有用信号尽量无失真放大，而对噪声尽量衰减。

衡量滤波器主要有 3 种技术指标：通带、阻带和过渡带。所谓通带，就是能使信号通过的频带，而抑制噪声通过的频带称为阻带。在通带中，理想滤波器的幅频特性 $H(\mathrm{j}\Omega)$（或 $H(\mathrm{j}\omega)$）为常数，并有线性相位，即 $\arg H(\mathrm{j}\Omega)=k\Omega$（或 $\arg H(\mathrm{j}\omega)=k\omega$）；在阻带中，要求 $H(\mathrm{j}\Omega)$（或 $H(\mathrm{j}\omega)$）接近于 0，对于相位没有要求；从通带到阻带之间有一个过渡带，理想状态下过渡带增益为 0，但在物理上不可实现，为此，要求过渡带越短越好。

模拟滤波器按幅度特性可分为低通滤波器、高通滤波器、带通滤波器和带阻滤波器，它们的理想幅度特性如图 5-1 所示。

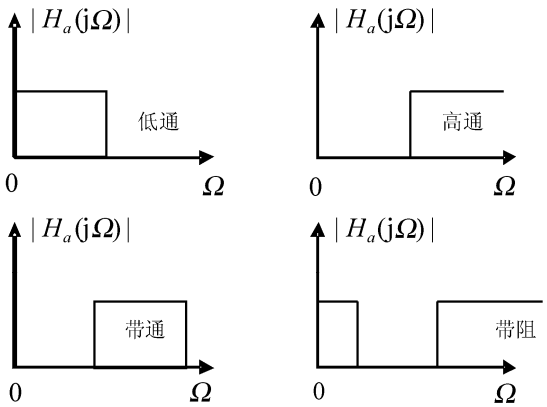


图 5-1　各种理想滤波器的幅频特性

1．模拟滤波器设计指标

如图 5-2 所示，模拟滤波器的设计指标主要有 R_P、Ω_P、R_S 和 Ω_S。其中 Ω_P 和 Ω_S 分别为通带截止频率和阻带截止频率；R_P 表示通带内的最大衰减系数，R_S 表示阻带内的最小衰减系数，单位为 dB，定义如下：

$$R_P=10\log\frac{|H_a(\mathrm{j}0)|^2}{|H_a(\mathrm{j}\Omega_P)|^2} \tag{5-1}$$

$$R_S = 10\log\frac{|H_a(\mathrm{j}0)|^2}{|H_a(\mathrm{j}\Omega_S)|^2} \tag{5-2}$$

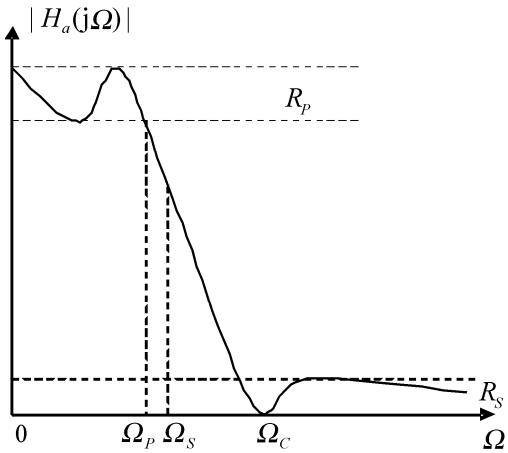


图 5-2　低通滤波器的设计指标示意图

此外，在滤波器设计中还有一个重要参数，即 3dB 截止频率 Ω_C，当 $|H_a(\mathrm{j}\Omega_c)|=1/\sqrt{2}$，则 $-20\log|H_a(\mathrm{j}\Omega_C)|=3\mathrm{dB}$。$\Omega_P$、$\Omega_S$ 和 Ω_C 的关系为

$$\Omega_P < \Omega_C < \Omega_S \tag{5-3}$$

给定滤波器技术指标后，进行滤波器设计，就是设计一个传输函数 $H_a(s)$，使它的幅度平方函数满足给定的指标 R_P 和 R_S，并通过 R_P、Ω_P、R_S 和 Ω_S，求出 $|H_a(\mathrm{j}\Omega)|^2$，从而得到所需的 $H_a(s)$。为了保证所设计的滤波器具有稳定性和最小相位，要求其传递函数的极点必须落在 S 平面的左半平面。

2．模拟滤波器设计方法

模拟滤波器设计方法已经相当成熟，并且有若干典型的模拟滤波器可供选择，如巴特沃思（Butterworth）滤波器、切比雪夫（Chebyshev）滤波器、椭圆滤波器和贝塞尔（Bessel）滤波器等，这些滤波器都有严格的设计公式，有现成的曲线和图表供设计人员选用。这些滤波器的特点各异，巴特沃思滤波器具有单调下降的幅频特性；切比雪夫滤波器的幅频特性在通带和阻带内有波动，可以提高选择性；贝塞尔滤波器通带内有较好的线性相位特性；椭圆滤波器的选择性相对于前三种滤波器是最好的。可以根据具体要求选用不同的滤波器。

在 MATLAB 中，提供了上述所有滤波器的设计函数，从而大大降低了滤波器的设计难度。设计模拟滤波器主要有两种方法。一种是首先调用 buttap、cheb1ap 等函数设计模拟低通滤波器原型，然后通过 lp2lp、lp2hp 等函数进行频率变换，将模拟低通滤波器原型转换为高通、带通和带阻等滤波器。另一种是根据滤波器性能参数，首先调用 buttord、cheb2ord 等阶数估计函数，对满足技术指标的滤波器的最低阶数进行估计，然后再调用滤波器完全设计函数进行模拟滤波器设计。

3．信号传输的条件

所谓信号无失真传输是指输入信号通过系统后，输出信号的幅值和输入信号的幅值成正比。允许有一定的延时，但没有波形上的畸变。因此，系统的频率响应 $H(\mathrm{j}\omega)$ 满足下面的特性：

$$\begin{cases}|H(\mathrm{j}\omega)|=k \\ \varphi(\omega)=\angle H(\mathrm{j}\omega)=-\omega t_d\end{cases} \tag{5-4}$$

式中：k、t_d 均为常数。

即信号无失真传输的条件是：系统的幅频响应 $|H(\mathrm{j}\omega)|$ 应为常数，相频响应 $\angle H(\mathrm{j}\omega)$ 应与频率 ω 成比例。或者说，滤波器应具有无限宽的定值幅频与线性相频。通常定义群延迟（group delay）为信号系统的延迟时间，即

$$t_d=\frac{\mathrm{d}\varphi(\omega)}{\mathrm{d}\varphi} \tag{5-5}$$

群延迟为相频特性曲线的斜率。对于信号无失真传输，t_d 为常数，即群延迟为常数；否则，它是频率 ω 的非线性函数。

5.2 模拟滤波器的原型设计

5.2.1 巴特沃思滤波器

巴特沃思模拟低通滤波器的平方幅频响应函数为

$$|H(\mathrm{j}\omega)|^2=A(\omega^2)=\frac{1}{1+(\omega/\omega_C)^{2N}} \tag{5-6}$$

式中：ω_C 为低通滤波器的截止频率；N 为滤波器的阶数。

巴特沃思滤波器的特点：通带内具有最大平坦的频率特性，且随着频率增大平滑单调下降；阶数越高，特性越接近矩形，过渡带越窄，传递函数无零点。

这里的特性接近矩形，是指通带频率响应段与过渡带频率响应段的夹角接近直角。通常该角为钝角，如果该角为直角，则为理想滤波器。

所谓滤波器的零点就是将该点的值代入传递函数后，传递函数的值为零。所谓函数的极点就是将该点的值代入传递函数后，传递函数的值为无穷大。滤波器的增益是指传递函数表达式前的常数。若系统的传递函数表示为

$$H(s)=\frac{Z(s)}{P(s)}=\frac{K(s-z(1))(s-z(2))\cdots(s-z(nz))}{(s-p(1))(s-p(2))\cdots(s-p(np))} \tag{5-7}$$

则滤波器零点为 $[z(1),z(2),\cdots,z(nz)]$，极点为 $[p(1),p(2),\cdots,p(np)]$，滤波器的增益为 K。

这里所说的零点和极点分布在一个圆上的拉普拉斯域中的形式，感兴趣的同学可查看数学中的拉普拉斯变换。

MATLAB 信号处理工具箱提供巴特沃思模拟低通滤波器原型设计函数巴特沃思，函数调用格式如下：

$$[z,p,k]=\mathrm{buttap}(n)$$

式中：n 为巴特沃思滤波器的阶数，z,p,k 分别为滤波器的零点、极点和增益。

巴特沃思滤波器的传递函数具有下面的形式：

$$H(s)=\frac{Z(s)}{P(s)}=\frac{K}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{5-8}$$

滤波器没有零点，极点为 $[p(1),p(2),\cdots,p(n)]$，滤波器的增益为 K。

在有关模拟滤波器设计的 MATLAB 程序中，经常遇到一些特定函数。$[b,a]=zp2tf(z,p,k)$ 为将模拟原型滤波器函数（如 buttap）设计出的零点 z，极点 p 和增益 k 形式转换为传递函数（transfer function）形式；其中，b 为滤波器传递函数分子多项式系数，a 为滤波器传递函数分母多项式系数。$[H,\omega]=freqs(b,a,\omega)$ 求出传递函数形式（分子和分母多项式的系数为 b,a）表示的滤波器对应于频率点 ω 的复数频率响应 H（包括实部和虚部），这里 ω 为一个向量，表示对应的角频率。若该函数不写输出变量，则执行后绘出该滤波器的幅频响应和相频响应图。

【例 5-1】设计产生一个 25 阶低通模拟滤波器原型，表示为零极点增益形式，并绘制频率特性图。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[Z,P,K]=buttap(25)
[num,den]=zp2tf(Z,P,K);
freqs(num,den);
```

运行程序，输出如下，效果如图 5-3 所示。

```
Z =
     []
P =
  -0.0628 + 0.9980i   -0.0628 - 0.9980i   -0.1874 + 0.9823i   -0.1874 - 0.9823i
  -0.3090 + 0.9511i   -0.3090 - 0.9511i   -0.4258 + 0.9048i   -0.4258 - 0.9048i
  -0.5358 + 0.8443i   -0.5358 - 0.8443i   -0.6374 + 0.7705i   -0.6374 - 0.7705i
  -0.7290 + 0.6845i   -0.7290 - 0.6845i   -0.8090 + 0.5878i   -0.8090 - 0.5878i
  -0.8763 + 0.4818i   -0.8763 - 0.4818i   -0.9298 + 0.3681i   -0.9298 - 0.3681i
  -0.9686 + 0.2487i   -0.9686 - 0.2487i   -0.9921 + 0.1253i   -0.9921 - 0.1253i   -1.0000
K =
     1.0000
```

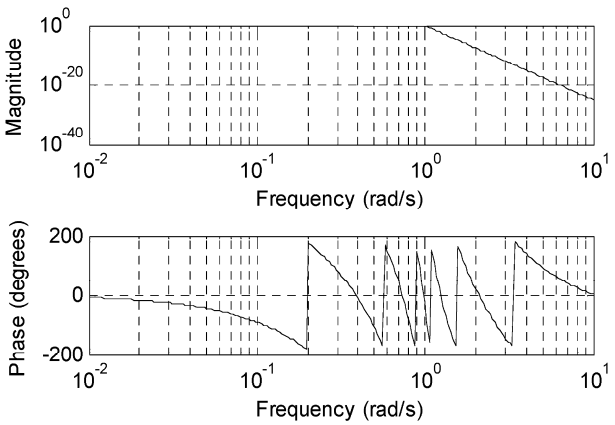


图 5-3　巴特沃思模拟滤波器特性图（n=25）

【例 5-2】设计模拟巴特沃思低通滤波器，并绘制幅频响应曲线。阶数分别为 3、5、10、15。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:0.01:2;
for i=1:4,
```

```
        switch i
            case 1;
                N=3;
            case 2;
                N=5;
            case 3;
                N=10;
            case 4;
                N=15;
        end;
    [z,p,k]=buttap(N);          %函数调用
    [b,a]=zp2tf(z,p,k);         %得到传递函数
    [h,w]=freqs(b,a,n);         %特性分析
    magh=abs(h);
    subplot(2,2,i);plot(w,magh);
    axis([0 2 0 1]);
    xlabel('w/wc');ylabel('|H(jw)|^2');
    title(['Butterworth analog filter N=',num2str(N)]);
    grid on;
    end
```

运行程序，效果如图 5-4 所示。

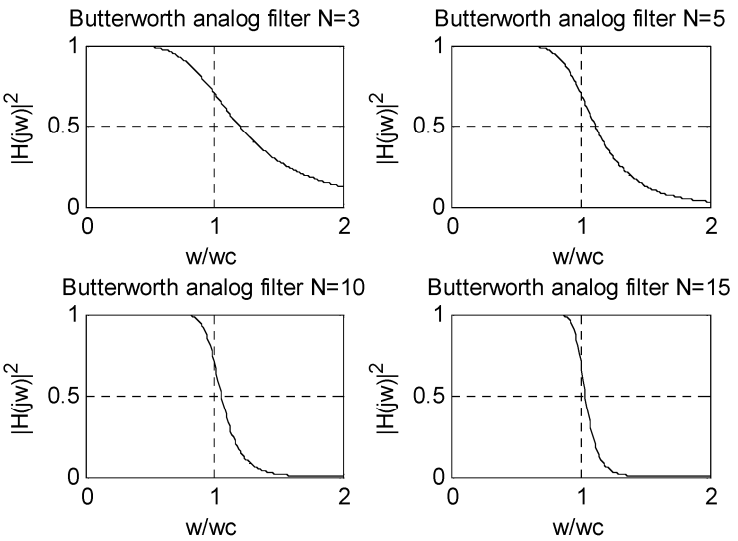


图 5-4　巴特沃思模拟低通滤波器的幅频特性曲线

由图 5-4 可以看出，巴特沃思滤波器的频率特性曲线无论在通带还是在阻带内，都是频率的单调函数。因此，当通带边界处满足指标要求，通带内有余量，可以将精度均匀地分布在整个通带内。为此，选择传递函数具有等波纹特性的滤波器，就能在阶数较低的情况下满足系统要求。

5.2.2　切比雪夫滤波器

切比雪夫滤波器有两种形式：幅频特性在通带内是等波纹，阻带内是单调的切比雪夫Ⅰ型

滤波器；幅频特性在通带内单调，阻带内是等波纹的切比雪夫Ⅱ型滤波器。

1．切比雪夫Ⅰ型滤波器

其平方幅值响应为

$$\left|H(\mathrm{j}\Omega)\right|^2=\frac{1}{1+\varepsilon^2C_N^2\left(\dfrac{\Omega}{\Omega_C}\right)} \tag{5-9}$$

该滤波器在通带内具有等波纹起伏特性，在阻带内则单调下降且具有更大的衰减。其中，ε 为小于 1 的正数，表示通带内幅度波动的程度，ε 越大波动幅度越大；Ω_C 为通带截止频率，N 为滤波器阶数，$C_N(x)$ 为 N 阶切比雪夫多项式，其定义为

$$C_N(x)=\begin{cases}\cos(N\arccos(x)), & |x|\leqslant 1\\ \cosh(N\,\mathrm{arch}(x)), & |x|>1\end{cases} \tag{5-10}$$

阶数 N 越高，幅频特性越接近矩形，其传递函数没有零点，所有极点分布在一椭圆上。

在 MATLAB 信号处理工具箱中，提供了 cheb1ap 函数实现该类滤波器，其调用格式如下：

[z,p,k] = cheb1ap(n,Rp)

其中：n 为滤波器阶数；z，p，k 分别为滤波器的零点、极点和增益；Rp 为通带波纹系数，单位为分贝（dB）。

利用该函数，可以设计 n 阶切比雪夫Ⅰ型模拟原型滤波器，滤波器传递函数为

$$H(s)=\frac{K}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{5-11}$$

【例 5-3】绘制切比雪夫Ⅰ型模拟低通滤波器的平方幅频响应曲线，阶数为 2，4，6，8。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:0.01:2;   %频率点
for i=1:4      %取 4 种滤波器
    switch i
        case 1, N=2;
        case 2; N=4;
        case 3; N=6;
        case 4; N=8;
    end
Rp=1;   %设置通滤波纹为 1dB
    [z,p,k]=cheb1ap(N,Rp);          %设计 Chebyshev Ⅰ型滤波器
    [b,a]=zp2tf(z,p,k);             %将零点极点增益形式转换为传递函数形式
    [H,w]=freqs(b,a,n);             %按 n 指定的频率点给出频率响应
    magH2=(abs(H)).^2;              %给出传递函数幅度平方
    posplot=['2,2',num2str(i)];     %将数字 i 转换为字符串,与'2,2'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
   title(['N=' num2str(N)]);        %将数字 N 转换为字符串并与'N='作为标题
   xlabel('w/wc');                  %显示横坐标
```

```
    ylabel('Chebyshev I |H(jw)|^2');   % 显示纵坐标
    grid on;
end
```

运行程序效果如图 5-5 所示。

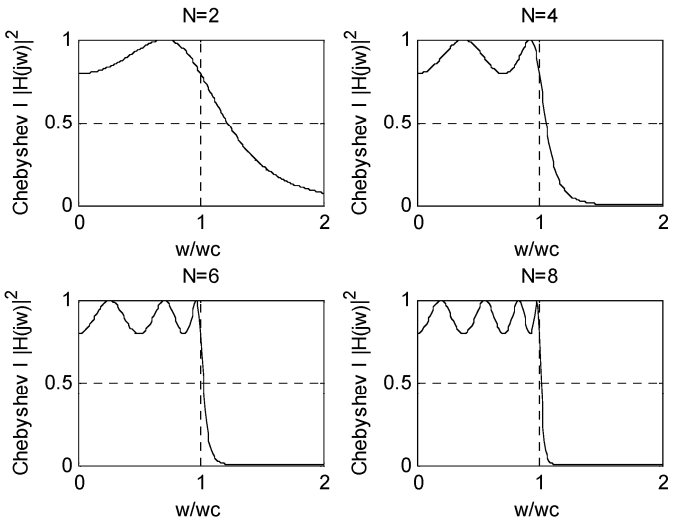


图 5-5　切比雪夫 I 型模拟原型滤波器平方幅频图

2．切比雪夫Ⅱ型滤波器

该滤波器平方幅值响应函数为

$$|H(\mathrm{j}\Omega)|^2=\frac{1}{\left[1+\varepsilon^2 C_N^2\left(\frac{\Omega}{\Omega_C}\right)\right]^{-1}} \tag{5-12}$$

滤波器在阻带内具有等波纹起伏特性，而在通带内则单调下降且具有更大的衰减特性：其参数意义同切比雪夫 I 型滤波器。该滤波器传递函数既有零点又有极点。

在 MATLAB 信号处理工具箱中，提供函数 cheb2ap 实现该类滤波器设计，其调用格式如下：

```
[z,p,k] = cheb2ap(n,Rs)
```

其中：n 为滤波器阶数；z、p、k 分别为滤波器的零点、极点和增益；Rs 为阻带内波纹峰值低于通带峰值的分贝数。

利用该函数，可以设计 *n* 阶切比雪夫 II 型模拟原型滤波器，滤波器传递函数为

$$H(s)=k\frac{(s-z(1))(s-z(2))\cdots(s-z(n))}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{5-13}$$

【例 5-4】分别设计 25 阶切比雪夫 I 型低通滤波器，通带内的最大衰减为 0.3dB，切比雪夫Ⅱ型低通模拟滤波器其阻带内的最小衰减为 Rs=50dB，并分别将切比雪夫 I 型，切比雪夫Ⅱ型低通模拟滤波器表示为传递函数形式和零极点增益形式。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[num,den]=cheb1ap(25,0.3)
[Z,P,K]=cheb2ap(25,50)
```

运行程序，输出如下：

```
num =
     []
>> den'
ans =
  Columns 1 through 5
  -0.0051-1.0013i   -0.0152-0.9855i   -0.0251-0.9542i   -0.0346-0.9078i   -0.0435-0.8471i
  Columns 6 through 10
  -0.0518-0.7731i   -0.0592-0.6868i   -0.0657-0.5897i   -0.0712-0.4833i   -0.0756-0.3693i
  Columns 11 through 15
  -0.0787-0.2495i   -0.0806-0.1257i   -0.0813           -0.0806+0.1257i   -0.0787+0.2495i
  Columns 16 through 20
  -0.0756+0.3693i   -0.0712+0.4833i   -0.0657+0.5897i   -0.0592+0.6868i   -0.0518+0.7731i
  Columns 21 through 25
-0.0435+0.8471i   -0.0346+0.9078i   -0.0251+0.9542i   -0.0152+0.9855i   -0.0051+ 1.0013i
Z =
          0 + 1.0020i          0 - 1.0020i          0 + 1.0180i          0 - 1.0180i
          0 + 1.0515i          0 - 1.0515i          0 + 1.1052i          0 - 1.1052i
          0 + 1.1844i          0 - 1.1844i          0 + 1.2978i          0 - 1.2978i
          0 + 1.4608i          0 - 1.4608i          0 + 1.7013i          0 - 1.7013i
          0 + 2.0757i          0 - 2.0757i          0 + 2.7165i          0 - 2.7165i
          0 + 4.0211i          0 - 4.0211i          0 + 7.9787i          0 - 7.9787i
P =
  Columns 1 through 5
-0.0154+0.9693i   -0.0473+0.9828i   -0.0829+1.0106i   -0.1253+1.0545i   -0.1790+1.1174i
  Columns 6 through 10
-0.2513+1.2033i   -0.3543+1.3183i   -0.5103+1.4689i   -0.7616+1.6589i   -1.1915+1.8689i
  Columns 11 through 15
-1.9452+1.9787i   -3.0900+1.5465i   -3.8335           -3.0900-1.5465i   -1.9452-1.9787i
  Columns 16 through 20
-1.191-1.8689i    -0.7616-1.6589i    -0.5103-1.4689i    -0.3543-1.3183i   -0.2513-1.2033i
  Columns 21 through 25
-0.1790-1.1174i    -0.1253-1.0545i   -0.0829-1.0106i    -0.0473-0.9828i   -0.0154-0.9693i
K =
    0.0791
```

【例 5-5】绘制切比雪夫Ⅱ型原型模拟滤波器的平方幅频响应曲线，阶数分别为 2，4，6，8。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:0.01:2;                          %频率点
for i=1:4                            %取 4 种滤波器
    switch i
        case 1, N=2;
        case 2; N=4;
        case 3; N=6;
        case 4; N=8;
    end
    Rs=16;   %设置通滤波纹为 16dB
```

```
    [z,p,k]=cheb2ap(N,Rs);          %设计切比雪夫Ⅱ型模拟原型滤波器
    [b,a]=zp2tf(z,p,k);             %将零点极点增益形式转换为传递函数形式
    [H,w]=freqs(b,a,n);             %按 n 指定的频率点给出频率响应
    magH2=(abs(H)).^2;              %给出传递函数幅度平方
    posplot=['2,2',num2str(i)];     %将数字 i 转换为字符串,与'2,2'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
  title(['N=' num2str(N)]);         %将数字 N 转换为字符串'N='合并作为标题
  xlabel('w/wc');                   %显示横坐标
  ylabel('Chebyshev II |H(jw)|^2');  % 显示纵坐标
  grid on;
end
```

运行程序效果如图 5-6 所示。可见切比雪夫Ⅱ型滤波器在通带内是单调平滑的，而阻带内却出现了波纹。随着滤波器阶数的增高，其幅频特性越接近矩形。

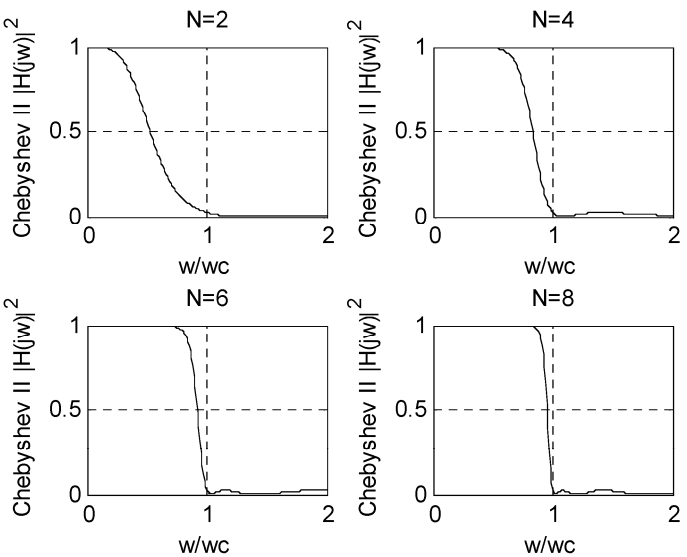


图 5-6　切比雪夫Ⅱ型模拟原型滤波器平方幅频图

5.2.3　贝塞尔滤波器

贝塞尔模拟低通滤波器的特点是在零频时具有最平坦的群延迟，并在整个通带内群延迟几乎不变。在零频时的群延迟为$\left(\frac{(2N)!}{2^N N!}\right)^{\frac{1}{N}}$。由于这一特点，贝塞尔模拟滤波器通带内保持信号形状不变。但数字贝塞尔滤波器没有平坦特性，因此 MATLAB 信号处理工具箱只有模拟贝塞尔滤波器设计函数。

函数 besselap 用于设计贝塞尔模拟低通滤波器原型，调用格式如下：

[z, p, k]=besselap(n)

其中：n 为滤波器阶数，不能大于 25；z、p、k 分别为滤波器的零点、极点和增益；该滤波器没有零点，返回的 z 为空矩阵。

利用该函数，可以设计 n 阶贝塞尔模拟原型滤波器，其传递函数为

$$H(s)=\frac{Z(s)}{P(s)}=\frac{k}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{5-14}$$

【例 5-6】设计模拟贝塞尔低通滤波器，并绘制幅频相应曲线。阶数分别为 5、10。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:0.01:2;
for i=1:2
    switch i
        case 1
            pos=1;      %设置极点
            N=5;
        case 2
            pos=3;
            N=10;
    end
    [z,p,k]=besselap(N);
    [b,a]=zp2tf(z,p,k);
    [h,w]=freqs(b,a,n);
magh2=(abs(h)).^2;
    phah=unwrap(angle(h));
    phah=phah*180/pi;
    subplot(2,2,pos);plot(w,magh2);
    axis([0 2 0 1]);
    xlabel('w/wc');ylabel('Bessel "H(jw)|^2');
    title(['N=',num2str(N)]);
    grid on;
    subplot(2,2,pos+1); plot(w,phah);
    xlabel('w/wc');ylabel('Bessel "Ph(jw)|');
    title(['N=',num2str(N)]);
    grid on;
end
```

运行程序，效果如图 5-7 所示。

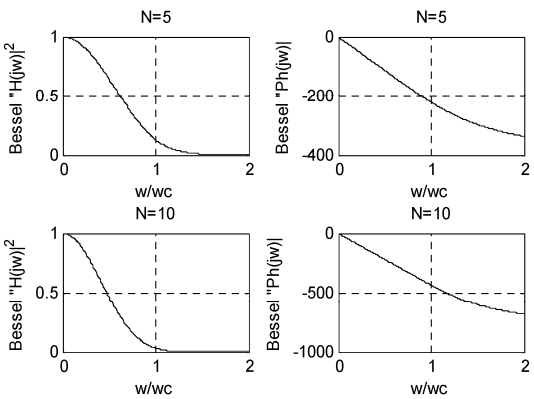


图 5-7　贝塞尔低通滤波器原型辅助相位特性图

5.2.4 椭圆滤波器

椭圆滤波器的平方幅值响应为

$$|H(\mathrm{j}\Omega)|^2=\frac{1}{1+u^2E_N^2(\Omega/\Omega_C)} \tag{5-15}$$

式中：u 为波纹系数，表示波纹情况；Ω_C 为通带截止频率，$E_N(\Omega/\Omega_C)$ 为椭圆函数。

椭圆滤波器在通带和阻带内均匀等波纹，巴特沃思和切比雪夫滤波器有更陡的下降斜度，但损失了通带和阻带内的波纹指标。在相同性能指标下，椭圆滤波器所需阶数最小，但相频特性具有明显的非线性。

在 MATLAB 信号处理工具箱中，提供函数 ellipap 实现该滤波器。其调用格式如下：

```
[z,p,k] = ellipap(n,Rp,Rs)
```

其中：z、p、k 分别为滤波器的零点、极点和增益；n 为滤波器阶数；Rp 为通带内波纹系数，单位为 dB；Rs 为阻带内波纹峰值低于通带峰值的分贝数。当阶数 n 为偶数时，z 和 p 的长度为 N，当 n 为奇数时，则 z 向量的长度为 N−1。

利用该函数，可以设计 n 阶椭圆滤波器，滤波器传递函数为

$$H(s)=k\frac{(s-z(1))(s-z(2))\cdots(s-z(n))}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{5-16}$$

【例 5-7】绘制 Elliptic 低通模拟原型滤波器的幅频平方响应曲线，阶数分别为 2，4，6，8。其实现的 MATLAB 程序如下：

```
>> clear all;
n=0:0.01:2;                          %频率点
for i=1:4                            %取 4 种滤波器
    switch i
        case 1, N=2;
        case 2; N=4;
        case 3; N=6;
        case 4; N=8;
    end
  Rp=1; Rs=15;                       %设置通滤波纹为 1dB,阻带衰减为 15dB
    [z,p,k]=ellipap(N,Rp,Rs);        %设计椭圆滤波器
    [b,a]=zp2tf(z,p,k);              %将零点极点增益形式转换为传递函数形式
    [H,w]=freqs(b,a,n);              %按 n 指定的频率点给出频率响应
    magH2=(abs(H)).^2;               %给出传递函数幅度平方
    posplot=['2,2',num2str(i)];      %将数字 i 转换为字符串,与'2,2'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
title(['N=' num2str(N)]);            %将数字 N 转换为字符串'N='合并作为标题
   xlabel('w/wc');                   %显示横坐标
   ylabel('椭圆|H(jw)|^2');          %显示纵坐标
   grid on;
end
```

运行程序效果如图 5-8 所示。可见阶数为 4 的椭圆滤波器的过渡带已相当窄（陡），但这种特性的获得是以牺牲通带和阻带的单调平滑特性为代价的。可以看到，滤波器的阶数越高，平

方幅频响应越接近于矩形。

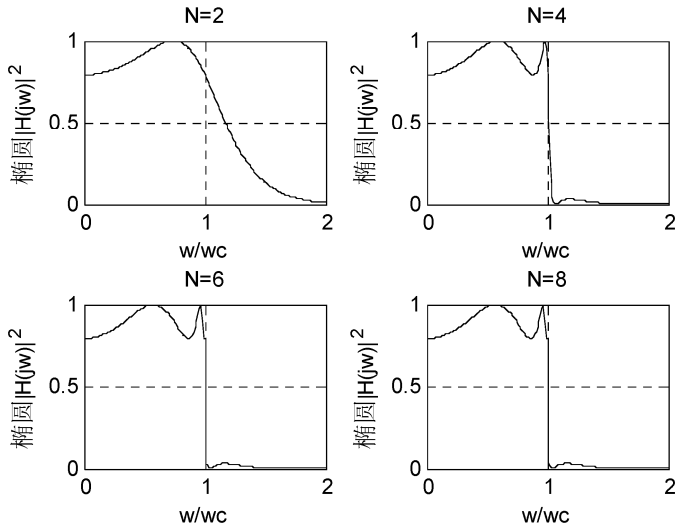


图 5-8　椭圆模拟原型滤波器平方幅频图

5.3　频率变换

前面所讲的模拟原型滤波器均是截止频率为 1 的滤波器，在实际设计中是很难遇到的，然而它是设计其他各类滤波器的基础。通常遇到的是截止频率任意的低通滤波器、高通滤波器、带通滤波器和带阻滤波器。如何由低通原型模拟滤波器设计这些滤波器呢？这就要用到频率变换。所谓频率变换是指各类滤波器（低通、高通、带通、带阻）和低通滤波器原型的传递函数中频率自变量之间的变换关系。通过频率变换，可以从模拟低通滤波器原型获得模拟的低通滤波器、高通滤波器、带通滤波器和带阻滤波器，再借助于 s 域至 z 域的变换关系，又可以设计各类后面将介绍的无限冲激响应数字滤波器，这是滤波器设计的重要方法之一。

MATLAB 信号处理工具箱有 lp2lp，lp2hp，lp2bp，lp2bs 四个频率变换函数，下面分别介绍。

1．lp2lp 函数

功能：用于实现由低通模拟原型滤波器至低通滤波器的频率变换。其调用格式如下：

[bt,at] = lp2lp(b,a,W0)

[At,Bt,Ct,Dt] = lp2lp(A,B,C,D,W0)

lp2lp 函数可将截止频率为 1rad/s 的模拟低通滤波器原型，转换为截止频率为 W0 的低通滤波器，这是利用 butter、cheby1、cheby2 和 ellip 函数设计数字滤波器的关键一步。

lp2lp 函数可以用传递函数和状态空间两种形式进行转换，但无论哪种形式，其输入必须是模拟滤波器原型。

当调用[bt, at]=lp2lp(b, a, W0)进行转换时，模拟滤波器表示形式如下：

$$H(s)=\frac{b(s)}{a(s)}=\frac{b(1)s^n+\cdots+b(n)s+b(n+1)}{a(1)s^m+\cdots+a(m)s+a(m+1)} \tag{5-17}$$

调用[At, Bt, Ct, Dt]=lp2lp(A,B,C,D,W0)进行转换时，模拟滤波器表示形式如下：

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \dot{\boldsymbol{y}} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}\end{aligned} \tag{5-18}$$

式中：$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 分别是状态转移方程矩阵。在状态空间中，该转换对应的转换矩阵为

$$\begin{aligned}\boldsymbol{A}_t &= W_0 \cdot \boldsymbol{A} \\ \boldsymbol{B}_t &= W_0 \cdot \boldsymbol{B} \\ \boldsymbol{C}_t &= \boldsymbol{C} \\ \boldsymbol{D}_t &= \boldsymbol{D}\end{aligned} \tag{5-19}$$

【例 5-8】将 4 阶椭圆模拟原型滤波器变换为截止频率为 0.5 的椭圆模拟低通滤波器，其中通带波纹 Rp=2dB，阻带衰减 Rs=30dB。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Rp=2;   Rs=30;                          %模拟原型滤波器的通带波纹与阻带衰减
[z,p,k]=ellipap(4,Rp,Rs);               %设计椭圆滤波器
[b,a]=zp2tf(z,p,k);                     %由零点极点增益形式转换为传递函数形式
n=0:0.01:2;
[h,w]=freqs(b,a,n);                     %给出复数频率响应
subplot(2,1,1);plot(w,abs(h).^2);       %绘出平方幅频函数
xlabel('w/wc');ylabel('椭圆|H(jw)|^2');
title('原型低通椭圆滤波器 (wc=1)');
grid on;
[bt,at]=lp2lp(b,a,0.5);                 %将模拟原型低通滤波器的截止频率变换为 0.5
[ht,wt]=freqs(bt,at,n);                 %给出复数频率响应
subplot(2,1,2);plot(wt,abs(ht).^2);     %绘出平方幅频函数
xlabel('w/wc');ylabel('椭圆|H(jw)|^2');
title('原型低通椭圆滤波器 (wc=0.5)');
grid on;
```

运行程序，效果如图 5-9 所示。

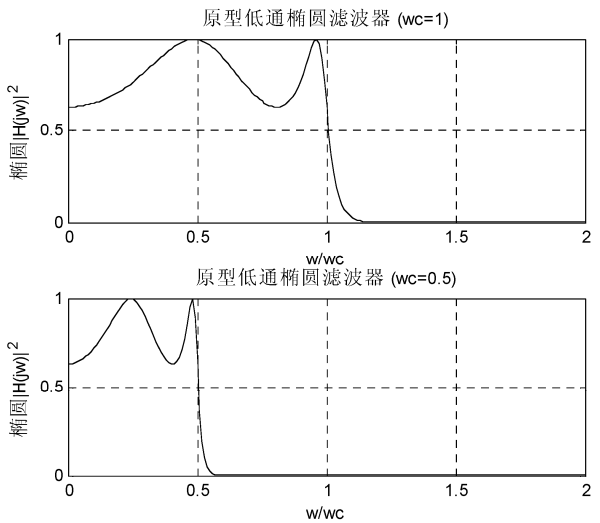


图 5-9　将 4 阶椭圆模拟原型滤波器变换为截止频率为 0.5 的椭圆模拟低通滤波器效果

2．lp2hp 函数

功能：用于实现由低通模拟滤波器至高通滤波器的频率变换。其调用格式如下：

[bt, at]=lp2hp(b, a, W0)

[At, Bt, Ct, Dt]=lp2hp(A, B, C, D, W0)

lp2hp 函数与 lp2lp 函数功能类似，只是将模拟低通滤波器原型转换为截止频率为 W0 的高通滤波器，这里不再说明。

该函数将模拟原型滤波器传递函数执行下面变换：

$$H(s)=H(p)\big|_{p=\frac{W_0}{s}} \tag{5-20}$$

【例 5-9】将 6 阶切比雪夫Ⅰ型原型滤波器变换为截止频率为 0.8 的模拟高通滤波器，其中通带波纹 Rp=0.5dB。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Rp=0.5;                          %设置滤波器的通切带波纹为 0.5dB
[z,p,k]=cheb1ap(6,Rp);           %设计切比雪夫 I 型模拟原型滤波器
[b,a]=zp2tf(z,p,k);              %由零点极点增益形式转换为传递函数形式
n=0:0.01:2;
[h,w]=freqs(b,a,n);              %给出复数频率响应
subplot(2,1,1);plot(w,abs(h).^2);  %绘出平方幅频函数
xlabel('w/wc');ylabel('椭圆|H(jw)|^2');
title('切比雪夫Ⅰ型低通原型滤波器  (wc=1)');
grid on;
[bt,at]=lp2hp(b,a,0.8);          %由低通原型滤波器转换为截止频率为 0.8 的高通滤波器
[ht,wt]=freqs(bt,at,n);          %给出复数频率响应
subplot(2,1,2);plot(wt,abs(ht).^2);  %绘出平方幅频函数
xlabel('w/wc');ylabel('椭圆|H(jw)|^2');
title('切比雪夫Ⅰ型高通滤波器  (wc=0.8)');
grid on;
```

运行程序，效果如图 5-10 所示。

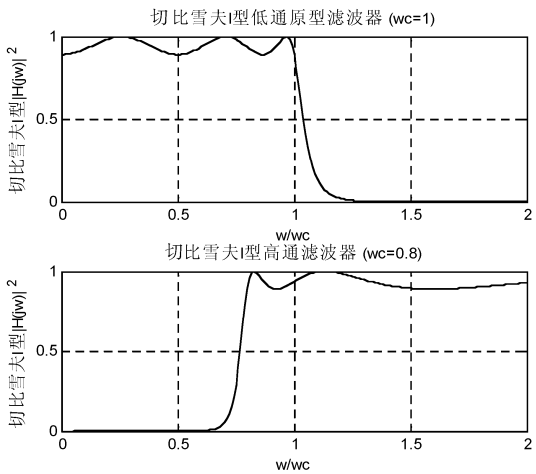


图 5-10　将 6 阶切比雪夫Ⅰ型原型滤波器变换为截止频率为 0.8 的模拟高通滤波器

3．lp2bp 函数

功能：用于实现由低通模拟原型滤波器至高通滤波器的转换。其调用格式如下：

[bt, at]=lp2bp(b, a, W0, Bw)

[At, Bt, Ct, Dt]=lp2bp(A,B,C,D,W0,Bw)

lp2bp 函数与 lp2lp 函数功能类似，只是将模拟低通滤波器原型转换为具有指定带宽 B_w 和中心频率为 W_0 的带通滤波器。

如果要求滤波器的低端截止频率为 w_1，高端截止频率为 w_2，则可先计算出 W_0 和 B_w：

$$\begin{aligned} W_0 &= \sqrt{w_1 \cdot w_2} \\ B_w &= w_2 - w_1 \end{aligned} \tag{5-21}$$

该函数将模拟原型滤波器传递函数执行下面的变换运算：

$$H(s) = H(p)\Big|_{p=\frac{W_0\left(\frac{s}{W_0}\right)^2+1}{B_w\frac{s}{W_0}}} \tag{5-22}$$

下面用例题说明截止频率和滤波器类型的转换。这里要注意，输出的带通滤波器阶数为模拟原型滤波器阶数的 2 倍。

【例 5-10】将 6 阶切比雪夫Ⅱ型原型滤波器变换为模拟带通滤波器，其中上边界截止频率为 0.8rad/s，下边界截止频率为 1.4rad/s，阻带误差 Rs=20dB。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Rs=20;                          %滤波器的阻带衰减为 20dB
[z,p,k]=cheb2ap(6,Rs);          %设计切比雪夫Ⅱ型模拟原型滤波器
[b,a]=zp2tf(z,p,k);             %由零点极点增益形式转换为传递函数形式
n=0:0.01:2;
[h,w]=freqs(b,a,n);             %给出复数频率响应
subplot(2,1,1);plot(w,abs(h).^2);   %绘出平方幅频函数
xlabel('w/wc');ylabel('切比雪夫 II 型|H(jw)|^2');
title('切比雪夫Ⅱ型低通原型滤波器 (wc=1)');
grid on;
w1=0.8; w2=1.4;                 %给定将要设计滤波器通带的下限和上限频率
w0=sqrt(w1*w2);                 %计算中心点频率
bw=w2-w1;                       %计算中心点频带宽度
[bt,at]=lp2bp(b,a,w0,bw);       %频率转换
[ht,wt]=freqs(bt,at,n);         %计算滤波器的复数频率响应
subplot(2,1,2);plot(wt,abs(ht).^2);   %绘出平方幅频函数
xlabel('w/wc');ylabel('切比雪夫Ⅱ型|H(jw)|^2');
title('切比雪夫Ⅱ型带通滤波器 (wc=0.8~1.4)');
grid on;
```

运行程序，效果如图 5-11 所示。

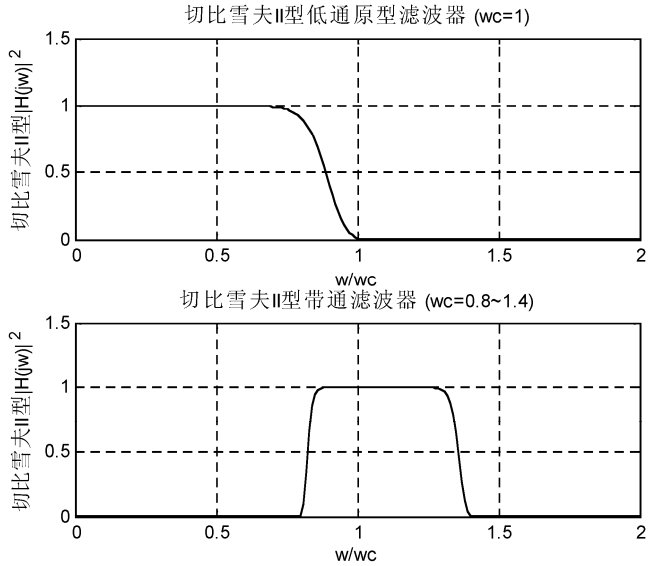


图 5-11　将 6 阶切比雪夫Ⅱ型原型滤波器变换为模拟带通滤波器

4．lp2bs 函数

功能：用于实现由低通模拟原型滤波器至带阻滤波器的频率变换。其调用格式如下：

[bt, at]=lp2bs(b, a, W0, Bw)

[At, Bt, Ct, Dt]=lp2bs(A, B, C, D, W0, Bw)

lp2bs 函数与 lp2bp 函数类似，其功能如上所示。该函数也可以用传递函数和状态空间两种形式进行转换，其输入必须是模拟滤波器原型。而

$$W_0 = \sqrt{w_1 w_2}, \quad B_w = w_2 - w_1 \tag{5-23}$$

式中：w_1 为带阻滤波器的下边界频率；w_2 为带阻滤波器上边界频率。若给定的边界频率单位为 Hz，需乘以 2π 。

该函数将模拟原型滤波器传递函数执行下面的变换运算：

$$H(s) = H(p)\Big|_{p=\frac{\frac{S}{W_0}}{B_w\left(\frac{S}{W_0}\right)^2+1}} \tag{5-24}$$

注意：输出的带阻滤波器和带通滤波器是滤波器原型阶数的 2 倍。

【例 5-11】将 6 阶巴特沃思原型滤波器变换为模拟带阻滤波器，其中上边界频率为 0.7rad/s，下边界频率为 1.5rad/s。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[z,p,k]=buttap(6);                    %设计巴特沃思模拟原型滤波器
[b,a]=zp2tf(z,p,k);                   %由零点极点增益形式转换为传递函数形式
n=0:0.01:2;
[h,w]=freqs(b,a,n);                   %给出复数频率响应
subplot(2,1,1);plot(w,abs(h).^2);     %绘出平方幅频函数
xlabel('w/wc');ylabel('巴特沃思|H(jw)|^2');
title('巴特沃思低通原型滤波器  (wc=1)');
```

```
grid on;
w1=0.7; w2=1.5;                    %给定将要设计带阻的下限和上限频率
w0=sqrt(w1*w2);                    %计算中心点频率
bw=w2-w1;                          %计算中心点频带宽度
[bt,at]=lp2bs(b,a,w0,bw);          %频率转换
[ht,wt]=freqs(bt,at,n);            %计算带阻滤波器的复数频率响应
subplot(2,1,2);plot(wt,abs(ht).^2);  %绘出平方幅频函数
xlabel('w/wc');ylabel('巴特沃思|H(jw)|^2');
title('巴特沃思带阻滤波器  (wc=0.7~1.5)');
grid on;
```

运行程序，效果如图 5-12 所示。

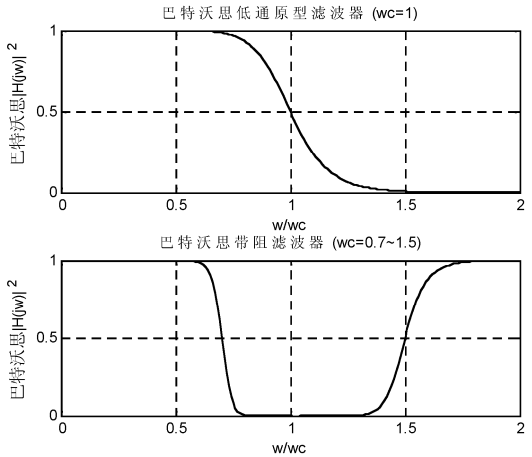


图 5-12　将 6 阶巴特沃思原型滤波器变换为模拟带阻滤波器

5.4　模拟滤波器离散化分析

5.4.1　冲激响应不变法分析

冲激响应不变法的基本原理是从滤波器的冲激响应出发，对具有传递函数 $G(s)$ 的模拟滤波器的冲激响应 $g(t)$，以周期 T 采样所得到的离散序列 $g(nT)$ 作为数字滤波器的冲激响应。在 MATLAB 信号处理工具箱中，提供了 impinvar 函数实现冲激响应不变法。

impinvar 函数功能：实现从模拟到数字的转换。

其调用格式如下：

[bz,az] = impinvar(b,a,fs)：把具有[b, a]模拟滤波器传递函数模型转换为采样频率为 fs 的数字滤波的传递函数模型[bz, az]。如果在函数中没有确定采样频率 fs 时，函数默认为 1Hz。

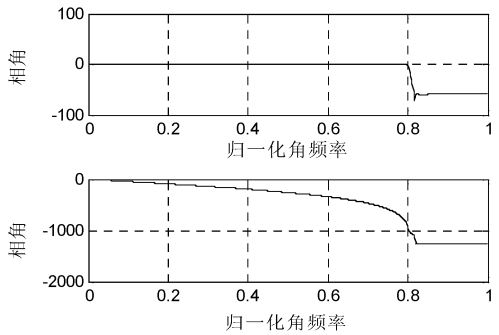
【例 5-12】运用冲激响应不变法设计一个低通椭圆数字滤波器，其通带上限临界频率是 600Hz，阻带临界频率是 620Hz，采样频率是 1500Hz，在通带内的最大衰减为 0.3dB，阻带内的最小衰减为 80dB。

实现的 MATLAB 程序代码如下：

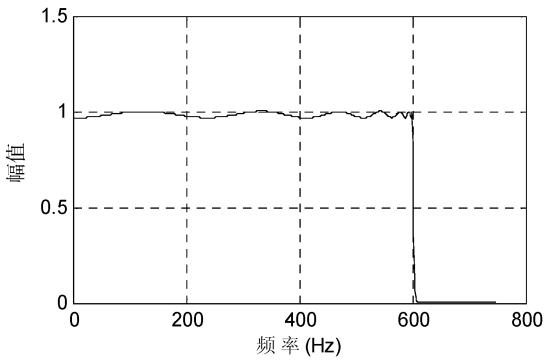
```
>> clear all;
wp=600*2*pi;
```

```
ws=620*2*pi;
rs=80;;rp=0.3;fs=1500;
[n,wn]=ellipord(wp,ws,rp,rs,'s');
[z,p,k]=ellipap(n,rp,rs);
[a,b,c,d]=zp2ss(z,p,k);
[at,bt,ct,dt]=lp2lp(a,b,c,d,wn);
[num1,den1]=ss2tf(at,bt,ct,dt);
[num2,den2]=impinvar(num1,den1,fs);
[h,w]=freqz(num2,den2);
figure;
winrect=[100,100,400,300];
set(gcf,'position',winrect);
set(gco,'linewidth',1);
freqz(num2,den2);
xlabel('归一化角频率');ylabel('相角');
figure;winrect=[100,100,400,300];
set(gcf,'position',winrect);
plot(w*fs/(2*pi),abs(h));
grid on;
xlabel('频率(Hz)');ylabel('幅值');
```

运行程序，效果如图 5-13 所示。



（a）角频率



（b）滤波特性

图 5-13　椭圆型数字滤波器特性

5.4.2 双线性变换法分析

为了克服冲激响应不变法产生的频率混叠现象，模拟向数字的转变通常又采用双线性变换法。

在 MATLAB 信号处理工具箱中，提供了 bilinear 函数实现双线性变换法。

bilinear 函数功能：双线性变换法。

其调用格式如下：

[zd,pd,kd] = bilinear(z,p,k,fs)：把模拟滤波器的零极点模型转换成数字滤波器的零极点模型，其中 fs 是采样频率。

[numd,dend] = bilinear(num,den,fs)：把模拟滤波器的传递函数模型转换成数字滤波器的传递函数模型，其中 fs 是采样频率。

[Ad,Bd,Cd,Dd] = bilinear(A,B,C,D,fs)：把模拟滤波器的状态方程模型转换成数字滤波器的状态方程模型，其中 fs 是采样频率。

以上三种滤波器可以另外限定预畸变频率 fp，在进行双线性变换之前，对采样频率进行畸变，以保证频率冲激响应在双线性变换前后具有良好的单值映射关系。具体的预畸变过程如下：

```
fp=2*pi*fp
fs=fp/tan(fp/fs/2)
```

例如，[zd,pd,kd] = bilinear(z,p,k,fs,fp)：在双线性变换之前应用了预畸变。

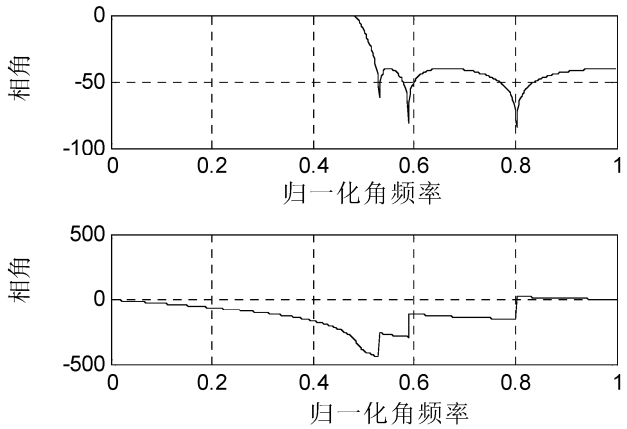
【例 5-13】设计一个数字信号处理系统，它的采样频率为 fs=1000Hz，希望在该系统中设计一个椭圆型高通数字滤波器，使其通带中允许的最小衰减为 0.3dB，阻带内的最小衰减为 40dB，通带上限临界频率为 300Hz，阻带下降临界频率为 350Hz。

其实现的 MATLAB 程序代码如下：

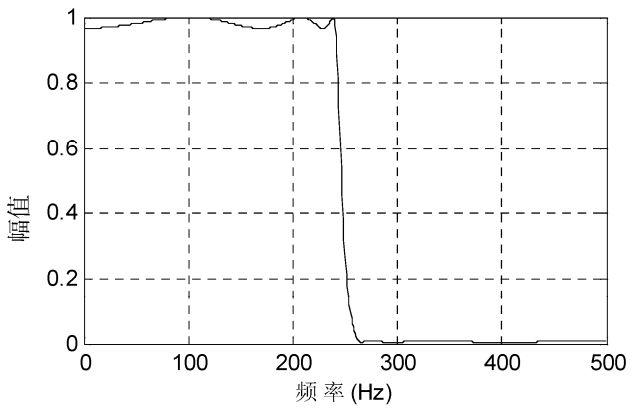
```
>> clear all;
wp=300*2*pi;
ws=350*2*pi;
rs=40;;rp=0.3;fs=1000;
[n,wn]=ellipord(wp,ws,rp,rs,'s');
[z,p,k]=ellipap(n,rp,rs);
[a,b,c,d]=zp2ss(z,p,k);
[at,bt,ct,dt]=lp2lp(a,b,c,d,wn);
[num1,den1]=ss2tf(at,bt,ct,dt);
[num2,den2]=bilinear(num1,den1,fs);
[h,w]=freqz(num2,den2);
figure;
winrect=[100,100,400,300];
set(gcf,'position',winrect);
set(gco,'linewidth',1);
freqz(num2,den2);
xlabel('归一化角频率');ylabel('相角');
figure;winrect=[100,100,400,300];
set(gcf,'position',winrect);
plot(w*fs/(2*pi),abs(h));
grid on;
```

```
xlabel('频率(Hz)');ylabel('幅值');
```

运行程序，效果如图 5-14 所示。



（a）角频率



（b）特性图

图 5-14　椭圆高通滤波器特性图

5.5　模拟滤波器的最小阶数选择

通过以上模拟滤波器的原型滤波器设计可以发现，随着阶数的增大，响应曲线在通带内越平缓，在阻带内的衰减越大。但是如何选择一个合适的滤波器阶数，使其刚好能实现系统最佳的性能，同时其实现不复杂呢？为此，在满足滤波器性能的前提条件下，滤波器阶数应越小越好。MATLAB 针对不同类型的滤波器，提供了最佳阶数选择计算函数，为滤波器设计进行最小阶数的计算。

5.5.1　巴特沃思模拟滤波器阶数选择函数

buttord 函数调用格式如下：

```
[n,Wn] = buttord(Wp,Ws,Rp,Rs)
[n,Wn] = buttord(Wp,Ws,Rp,Rs,'s')
```

该函数能返回符合技术指标要求的滤波器最小阶数 n，以及巴特沃思滤波器的固有频率 Wn

（即 3B 频率）。其中，Rp 为通带内的衰减；Rs 为阻带内的衰减；Wp 为通带截止频率；Ws 为阻带截止频率；'s'该参数表示设计的滤波器为模拟滤波器。

需要注意的是，此处的 Wp、Ws 是归一化频率，范围为[0 1]，对应π弧度。所谓归一化频率，在 MATLAB 中，通常使用 1/2 采样频率进行归一化处理。例如，对于采样频率为 1000Hz 的系统，则 300Hz 的归一化为 300/500=0.6。如果要归一化频率转化为角频率，则需归一化频率乘以π；如果要转换为 Hz，则需将归一化频率乘以采样频率的 1/2。

此外，还需要说明的是，对于不同类型的滤波器，Wp 和 Ws 使用不同的数据格式。

【例 5-14】buttord 函数用法示例。

```
>> clear all;
Wp = [60 200]/500; Ws = [50 250]/500;
Rp = 3; Rs = 40;
[n,Wn] = buttord(Wp,Ws,Rp,Rs)
[b,a] = butter(n,Wn);
freqz(b,a,128,1000);
title('n=16  巴特沃思滤波器');
```

运行程序，输出如下，效果如图 5-15 所示。

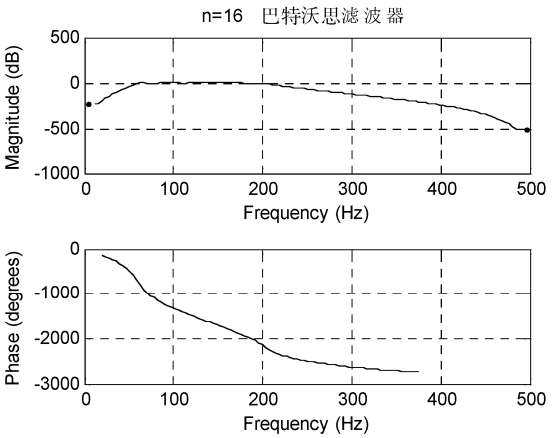


图 5-15　巴特沃思滤波器频率特性

```
n =
     16
Wn =
    0.1198    0.4005
```

5.5.2　切比雪夫Ⅰ型模拟滤波器阶数选择函数

cheb1ord 函数调用格式如下：

```
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs)
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs,'s')
```

该函数返回符合要求的滤波器最小阶数，以及切比雪夫Ⅰ型滤波器固有频率Wn，其参数意义同 buttord 函数。

【例 5-15】cheb1ord 函数用法示例。

```
>> clear all;
Wp = [60 200]/500; Ws = [50 250]/500;
Rp = 3; Rs = 40;
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs)
[b,a] = cheby1(n,Rp,Wp);
freqz(b,a,512,1000);
title('n=7 切比雪夫Ⅰ型滤波器');
```

运行程序，输出如下，效果如图 5-16 所示。

```
n =
     7
Wp =
    0.1200    0.4000
```

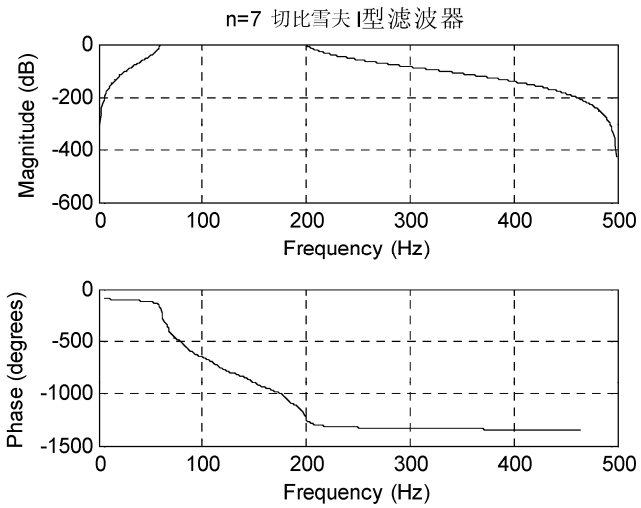


图 5-16　切比雪夫Ⅰ型滤波器频率特性

5.5.3　切比雪夫Ⅱ型模拟滤波器阶数选择函数

cheb2ord 函数调用格式如下：

[n,Ws] = cheb2ord(Wp,Ws,Rp,Rs)

[n,Ws] = cheb2ord(Wp,Ws,Rp,Rs,'s')

该函数返回符号要求的滤波器最小阶数 N 以及椭圆滤波器固有频率 Wn，其参数意义与函数 buttord 相同。

需要注意的是，如果阻带内的最大衰减 Rs 远大于通带内的最小衰减 Rp，或者通带截止频率 Wp 与阻带截止频率 Ws 十分接近，则由于数值精度的限制，阶数将趋于无穷大。

【例 5-16】cheb2ord 函数用法示例。

```
>> clear all;
Wp = [60 200]/500; Ws = [50 250]/500;
Rp = 3; Rs = 40;
[n,Ws] = cheb2ord(Wp,Ws,Rp,Rs)
[b,a] = cheby2(n,Rs,Ws);
```

```
freqz(b,a,512,1000)
title('n=7 切比雪夫Ⅱ型滤波器');
```

运行程序，输出如下，效果如图 5-17 所示。

```
n =
     7
Ws =
    0.1000    0.5000
```

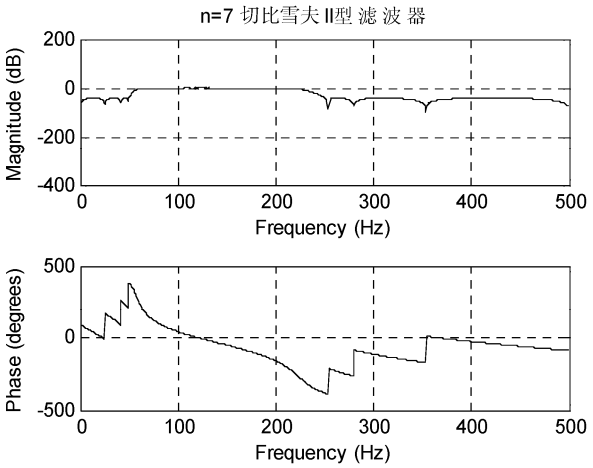


图 5-17　切比雪夫Ⅱ型滤波器频率特性

5.6　模拟滤波器的性能测试

滤波器设计好之后，一般要进行各方面的测试。在正式设计滤波器之前，先介绍如何测试滤波器的性能。对于模拟滤波器，在前面已经采用函数 freqs 来求模拟滤波器的频率响应，这里详细介绍该函数，其调用格式如下：

```
h = freqs(b,a,w)
[h,w] = freqs(b,a,n)
```

式中：b，a 分别为模拟滤波器传递函数分子和分母多项式系数；h 为对应频率点的传递函数值；w 为频率点的值；n 为频率点数。若 n=128，则用 128 个频率点来给出此模拟滤波器的频率特性（给定频率点的传递函数值），默认时为 200。若该函数不写输出变量，则执行后绘出该滤波器的幅频响应和相位响应图。此函数模拟滤波器的传递函数形式为

$$H(s)=\frac{B(s)}{A(s)}=\frac{b(1)s^{nb}+b(2)s^{nb-1}+\cdots+b(nb+1)}{a(1)s^{na}+a(2)s^{na-1}+\cdots+a(na+1)} \tag{5-25}$$

MATLAB 工具箱还提供了 abs 和 angle 两个函数，由频率响应 $H(\mathrm{e}^{\mathrm{j}\omega})$ 求幅频响应 $\left|H(\mathrm{e}^{\mathrm{j}\omega})\right|$ 和相频响应 $\angle H(\mathrm{e}^{\mathrm{j}\omega})$。其中 angle 的输出单位为 rad。可采用 rad2deg 函数转化为度。另外注意函数的幅频响应经常用分贝（dB）来表示。

求出的幅频响应 $\left|H(\mathrm{e}^{\mathrm{j}\omega})\right|$ 通过下式转换为分贝：20lg10($H(\mathrm{e}^{\mathrm{j}\omega})$)（dB）。

【例 5-17】已知模拟滤波器的传递函数为 $H(s)=\dfrac{0.2s^2+0.3s+1}{s^2+0.4s+1}$，绘制滤波器的幅频响应和相频响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a = [1 0.4 1];                    %滤波器传递函数分母多项式系数
b = [0.2 0.3 1];                  %滤波器传递函数分子多项式系数
w = logspace(-1,1);
freqs(b,a,w)
h = freqs(b,a,w);
mag = abs(h);
phase = angle(h);
subplot(2,1,1), loglog(w,mag)          %运用双对数坐标绘制幅频响应
grid on;
xlabel('角频率');ylabel('振幅');
subplot(2,1,2), semilogx(w,phase)      %运用半对数坐标绘相频响应
grid on;
xlabel('角频率');ylabel('相位');
```

运行程序，效果如图 5-18 所示。

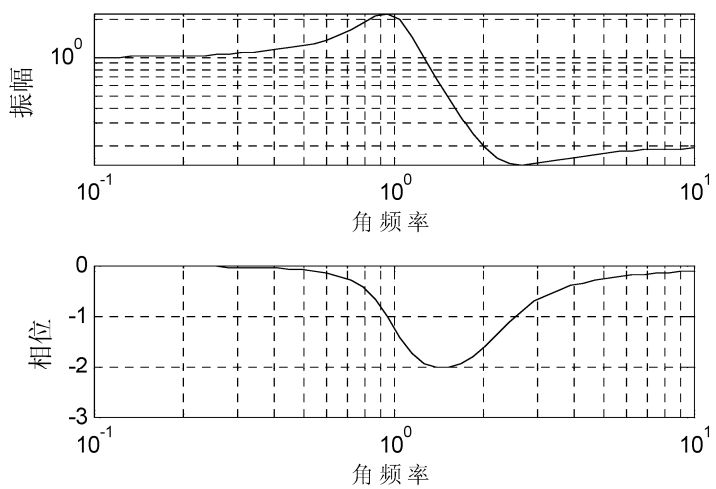


图 5-18　滤波器的幅频响应和相频响应

我们知道，除了用传递函数描述滤波器特性外，还可用脉冲（冲激）响应来描述滤波器，因为在模拟滤波器中，脉冲响应与传递函数是拉普拉斯变换对。此外还可以用阶跃响应（输入一个阶跃时系统的输出）来描述滤波器特性。下面介绍在 MATLAB 中如何得到模拟滤波器的脉冲响应和阶跃响应。

将滤波器的传递函数表示成分子和分母多项式系数的形式，如分子和分母多项式的系数为[b, a]，则在 MATLAB 中用 H=[tf(b, a)]来表示此模拟滤波器，采用[y, t]=impulse(H)给出该系统的模拟脉冲响应。采用[y, t]=step(H)来得到该系统的阶跃响应。这两个函数与 freqs 一样，若没有输出则程序自动绘图模拟滤波器的脉冲响应或阶跃响应。输出值 y，t 分别为该滤波器的脉冲响应或阶跃响应及其对应的时间序列。

另外，还可以运用一个输入信号模拟该滤波器的输出。若给定滤波器的输入值序列和对应的时间序列为 u 和 t，则系统的输出可用 y=lsim(H, u, t)来模拟，y 为对应 t 的输出。若该函数没有输出变量则程序自动绘图显示。

【例 5-18】设计一个 5 阶的切比雪夫Ⅰ型带通滤波器，通带波纹 3dB，下边界频率 100Hz，上边界频率 500Hz，绘制幅频响应图。给出该滤波器的脉冲响应、阶跃响应。假定输入 sin(2*pi*30*t)+0.5*cos(2*pi*300*t)+2*sin(2*pi*800*t)的信号，求模拟滤波器的输出并给出模拟输入信号和模拟输出信号的傅里叶振幅谱。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=5; Rp=3;                              %滤波器阶数
f1=100;f2=500;                          %滤波器的边界界限(Hz)
w1=2*pi*f1;   w2=2*pi*f2;               %边界频率(rad/s)
[z,p,k]=cheb1ap(N,Rp);                  %设计切比雪夫 Ⅰ型原型低通滤波器
[b,a]=zp2tf(z,p,k);                     %转换为传递函数形式
Wo=sqrt(w1*w2);                         %中心频率
Bw=w2-w1;                               %频带宽度
[bt,at]=lp2bp(b,a,Wo,Bw);               %频率转换
[h,w]=freqs(bt,at);                     %计算复数频率响应
figure;
subplot(2,2,1);semilogy(w/2/pi,abs(h));  %绘制幅频特性
xlabel('频率/Hz'); title('幅频图');
grid on;
subplot(2,2,2);plot(w/2/pi,angle(h)*180/pi);  %绘制相频响应
xlabel('频率/Hz');ylabel('相位图/^o');title('相频图');
grid on;
H=[tf(bt,at)];                          %在 MATLAB 中表示此滤波器
[h1,t1]=impulse(H);                     %绘出系统的脉冲响应图
subplot(2,2,3);plot(t1,h1);
xlabel('时间/s');title('脉冲响应');
[h2,t2]=step(H);                        %绘出系统的阶跃响应图
subplot(2,2,4);plot(t2,h2);
xlabel('时间/s');title('阶跃响应');
figure;
dt=1/2000;
t=0:dt:0.1;                             %给出模拟滤波器输出的时间范围
% 模拟输入信号
u=sin(2*pi*30*t)+0.5*cos(2*pi*300*t)+2*sin(2*pi*800*t);
subplot(2,2,1);plot(t,u)                %绘制模拟输入信号
xlabel('时间/s');title('输入信号');
[ys,ts]=lsim(H,u,t);                    %模拟系统的输入 u 时的输出
subplot(2,2,2);plot(ts,ys);             %绘制模拟输入信号
xlabel('时间/s');title('输出信号');
% 绘制输入信号振幅谱
subplot(2,2,3);plot((0:length(u)-1)/(length(u)*dt),abs(fft(u))*2/length(u));
xlabel('频率/Hz');title('输入信号振幅谱');
subplot(2,2,4);
Y=fft(ys);
%绘制输出信号振幅谱
plot((0:length(Y)-1)/(length(Y)*dt),abs(Y)*2/length(Y));
xlabel('频率/Hz');title('输出信号振幅谱');
```

运行程序，效果如图 5-19 和图 5-20 所示。

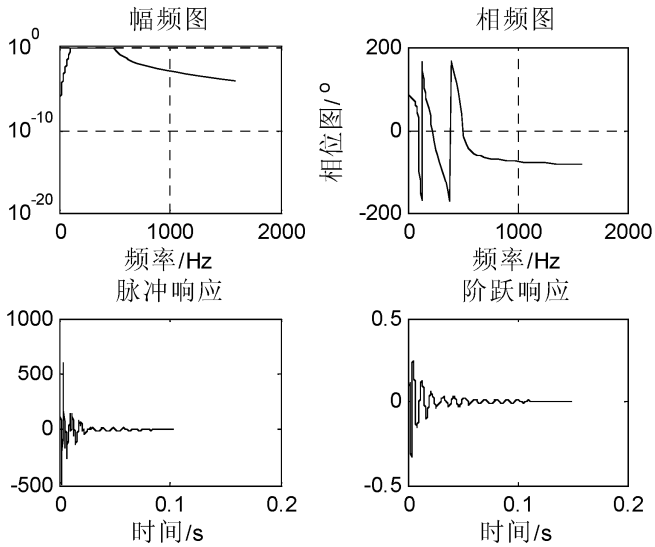


图 5-19　设计的切比雪夫Ⅰ型滤波器的幅频响应、相频响应、脉冲响应和阶跃响应

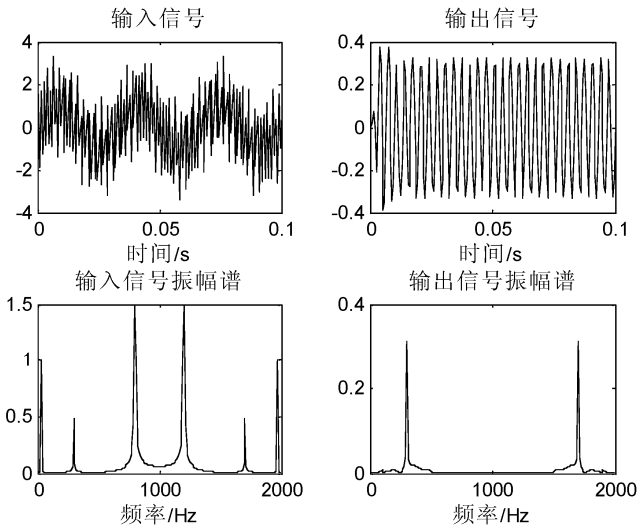


图 5-20　设计滤波器模拟输入和输出信号的时间域图形和振幅谱

图 5-19 给出了该程序得到的幅频图、相频图、脉冲响应和阶跃响应。幅频图清楚地给出了通带范围和阻带范围。图 5-20 给出了模拟输入、输出信号的时间域波形及振幅谱。输入信号中含有 30Hz，300Hz 和 500Hz 的波，由滤波器的幅频特性（图 5-19 的幅频图）可知，30Hz 和 500Hz 全在阻带内，只有 300Hz 的波可通过滤波器。从输入信号和模拟输出信号的时间域波形和振幅谱，可以看到达到了预期的效果。注意，由该滤波器的相频特性（图 5-19 的相频图）可知，该滤波器并不是线性相位的。

5.7 模拟滤波器的设计

5.7.1 模拟滤波器设计步骤

用户对设计的滤波器提出设计要求，可以针对滤波器的设计要求设计滤波器。通常用户对模拟滤波器提出的要求有：

- 滤波器的性能指标，包括截止频率 ω_0（对于低通和高通）或上下边界频率 ω_1、ω_2，通带波纹、阻带衰减等。
- 滤波器的类型，通常为巴特沃思、切比雪夫Ⅰ、切比雪夫Ⅱ、Elliptic 或贝塞尔滤波器。

根据滤波器的类型，通常按下列步骤设计滤波器：

（1）给定模拟滤波器的性能指标，如截止频率 ω_0（对于低通和高通）或上下边界频率 ω_1、ω_2，通带波纹、阻带衰减以及滤波器类型等（用户给定）。

（2）确定滤波器阶数。

（3）设计模拟低通滤波器。MATLAB 信号处理工具箱的滤波器原型设计函数有 butterap、cheb1ap、cheb2ap、ellipap、besselap。

（4）按频率变换设计模拟滤波器（低通、高通、带通、带阻）。MATLAB 信号处理工具箱的频率变换函数有 lp2lp、lp2hp、lp2bp、lp2bs。

【例 5-19】设计一个巴特沃思模拟带通滤波器，技术指标为：W1=2000Hz，W2=3000Hz，两侧的过渡带宽为 500Hz，通带波纹 Rp=1dB，阻带衰减 Rs=60dB。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=[2000 3000]*2*pi;                    %性能指标
ws=[1500 3500]*2*pi;
Rp=1;
Rs=60;
[N,Wn]=buttord(wp,ws,Rp,Rs,'s')         %估计滤波器的最小阶数 N
[b,a]=butter(N,Wn,'s');                 %利用 butter 函数设计滤波器
% output Am response
w=linspace(1,4000,1000)*2*pi;
H=freqs(b,a,w);
plot(w/(2*pi),20*log10(abs(H)));
xlabel('Frequency(Hz)');
ylabel('Magnitude(dB)');
title(['Butterworth Analog bandpass filter order N=',num2str(N)]);
grid on;
```

运行程序输出如下，效果如图 5-21 所示。

```
N =      14
Wn =   1.0e+004 *
       1.2342       1.9192
```

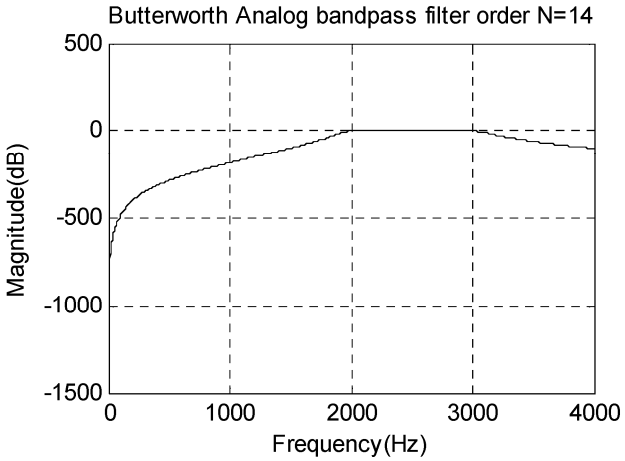


图 5-21　巴特沃思带通模拟滤波器幅频特性图

5.7.2　模拟滤波器设计函数

上面滤波器的设计频率比较麻烦，根据设计要求求解滤波器的最小阶数和边界频率之后需要设计模拟原型滤波器并进行频率转换。其实 MATLAB 将这一系列的过程组合成了更为方便的设计函数：butter，cheby1，cheby2，ellip，besself。这些函数称为模拟滤波器完全设计函数。用户在求得滤波器的最小阶和截止频率之后只需调用一次完全设计函数就可以自动完成所有设计过程，编程十分简单。这些工具函数适用于模拟滤波器的设计，但同样也适用于数字滤波器。

下面分别介绍这些函数的用法。

1．butter 函数

功能：设计巴特沃思模拟滤波器。其调用格式如下：

[b,a]=butter(n, Wn, 's')：设计 n 阶模拟低通滤波器，截止频率为 Wn rad/s，该函数返回长度为 n+1 的 A、B 两处向量，表示的系统传递函数为

$$H(s)=\frac{B(s)}{A(s)}=\frac{b_1s^n+b_2s^{n-1}+\cdots+b_{n+1}}{s^n+a_2s^{n-1}+\cdots+a_{n+1}}$$

[A, B, C, D]=butter(...)：返回的是状态转换矩阵 A、B、C 和 D。

如果 Wn 是一个有两个元素的向量，即 Wn=[W1 W2]（W1<W2），则 butter(n, Wn,'s')返回 2n 阶模拟带通滤波器，通带范围是 W1<W<W2。

[z,p,k]=butter(n, Wn, 'ftype', 's')：返回 n 列或者 2n 列的零极点向量和增益 K。

ftype 代表滤波器类型，可取如下几个参数：

（1）'high'：代表所设计的是高通滤波器。

（2）'stop'：代表所设计的是带阻滤波器。

（3）'s'：代表是模拟滤波器设计。

[b, a]=butter(n, Wn)和[b,a]=butter(n, Wn, 'ftype')：用于数字滤波器的设计，该函数与 buttord 联合使用可设计最小阶数的巴特沃思模拟滤波器。

2．cheby1 函数

功能：设计切比雪夫Ⅰ型模拟滤波器。其调用格式如下：

[b, a]=cheby1(n, Rp, Wn)
[b, a]=cheyb1(n, Rp, Wn, 'ftype')
[b, a]=cheby1(n, Rp, Wn, 's')
[b, a]=cheby1(n, Rp, Wn, 'ftype' 's')
[z, p,k]=cheby1(...)
[A, B, C, D]=cheby1(...)

该函数用于设计切比雪夫Ⅰ型低通、高通、带通和带阻模拟或数字滤波器。通带内，该滤波器具有等波纹特性，阻带内单调下降。切比雪夫Ⅰ型滤波器较切比雪夫Ⅱ型的阻带衰减更快，但通带内的误差加大。

函数引用中，Rp 为通带波纹系数，其余参数与 butter 函数的参数意义相同。

3．cheby2 函数

功能：设计切比雪夫Ⅱ型模拟滤波器。其调用格式如下：

[b, a]=cheby2(n, Rp, Wn)
[b, a]=cheyb2(n, Rp, Wn, 'ftype')
[b, a]=cheby2(n, Rp, Wn, 's')
[b, a]=cheby2(n, Rp, Wn, 'ftype' 's')
[z, p,k]=cheby2(...)
[A, B, C, D]=cheby2(...)

该函数用于设计切比雪夫Ⅱ型低通、高通、带通和带阻模拟或者数字滤波器。Rs 为阻带最低衰减系数，其余参数与 butter 函数相同。该函数的 MATLAB 使用方法和引用格式与 cheby1 函数完全相同，在此不再介绍。

4．ellip 函数

功能：设计椭圆滤波器。其调用格式如下：

[b,a]=ellip(n, Rp,Rs, Wn)
[b, a]=ellip(n, Rp, Rs, Wn, 'ftype')
[b, a]=ellip(n, Rp, Rs, Wn, 's')
[b, a]=ellip(n, Rp, Rs, Wn, 'ftype', 's')
[z, p, k]=ellip(...)
[A, B, C, D]=ellip(...)

该函数用于设计椭圆型低通、高通、带通和带阻模拟或者数字滤波器。椭圆滤波器在通带和阻带内均为等波纹，巴特沃思和切比雪夫滤波器有更陡的下降斜度，但会损失通带和阻带内的波纹指标，在相同的性能指标下椭圆滤波器所需阶数最小。

Rp 为通带波纹系数（dB），Rs 为阻带衰减系数（dB），其他参数同 butter 函数。

5．besself 函数

功能：设计贝塞尔模拟滤波器。其调用格式如下：

[b, a]=besself(n, Wn)
[b, a]=besself(n, Wn, 'ftype')
[z, p, k]=besself(...)
[A, B, C, D]=besself(...)

该函数仅用于设计贝塞尔低通、高通、带通和带阻模拟滤波器。滤波器在通带内有恒定的群延迟特性，可以较好地保持通带内的信号波形。其参数意义与 butter 函数相同。

贝塞尔模拟滤波器与巴特沃思、切比雪夫和椭圆滤波器相比，过渡带下降幅度最慢，因此，该函数设计滤波器时所需滤波器阶数一般比较大。

【例 5-20】利用 butter 设计一个模拟低通滤波器。

```
>> d = fdesign.lowpass; designopts(d,'butter')
ans =
      FilterStructure: 'df2sos'
          SOSScaleNorm: ''
          SOSScaleOpts: [1x1 fdopts.sosscaling]
          MatchExactly: 'stopband'
>> hd = design(d,'butter','matchexactly','stopband')
hd =
              FilterStructure: 'Direct-Form II, Second-Order Sections'
                    Arithmetic: 'double'
                     sosMatrix: [13x6 double]
                   ScaleValues: [14x1 double]
       OptimizeScaleValues: true
           PersistentMemory: false
>> d = fdesign.highpass('n,fc',8,.6); design(d,'butter')
```

运行程序，效果如图 5-22 所示。

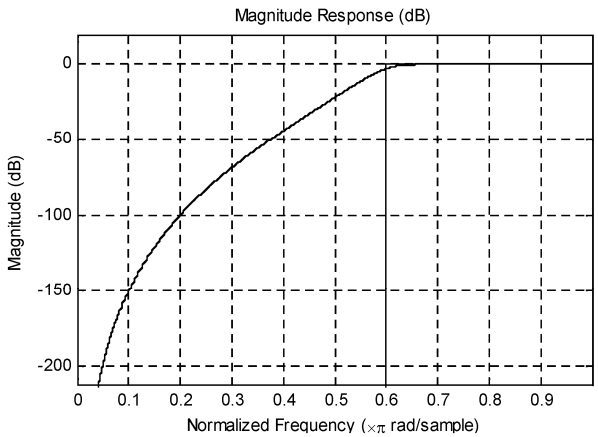


图 5-22　butter 函数设计的滤波器频率特性效果图

【例 5-21】利用 cheby1 设计一个模拟低通滤器。

```
>> d = fdesign.lowpass; designopts(d,'cheby1')
ans =
      FilterStructure: 'df2sos'
          SOSScaleNorm: ''
          SOSScaleOpts: [1x1 fdopts.sosscaling]
          MatchExactly: 'passband'
>> hd = design(d,'cheby1','matchexactly','passband')
 hd =
              FilterStructure: 'Direct-Form II, Second-Order Sections'
                    Arithmetic: 'double'
                     sosMatrix: [5x6 double]
```

```
                    ScaleValues: [6x1 double]
          OptimizeScaleValues: true
              PersistentMemory: false
>> d = fdesign.highpass('n,fp,ap',7,20,.4,50);
hd = design(d,'cheby1')
hd =
                FilterStructure: 'Direct-Form II, Second-Order Sections'
                      Arithmetic: 'double'
                      sosMatrix: [4x6 double]
                    ScaleValues: [5x1 double]
          OptimizeScaleValues: true
              PersistentMemory: false
>> fvtool(hd)                          %利用 fvtool 函数设计滤波器频率特性，效果如图 5-23 所示
```

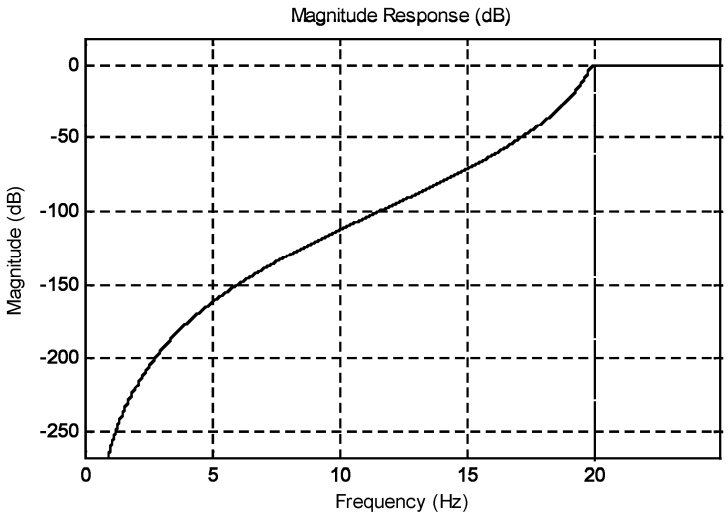


图 5-23　cheby1 函数设计的滤波器频率特性效果图

【例 5-22】利用 cheby2 设计一个模拟低通滤波器。

```
>> d = fdesign.lowpass;
hd = design(d,'cheby2','matchexactly','passband')
hd =
                FilterStructure: 'Direct-Form II, Second-Order Sections'
                      Arithmetic: 'double'
                      sosMatrix: [5x6 double]
                    ScaleValues: [6x1 double]
          OptimizeScaleValues: true
              PersistentMemory: false
>> d = fdesign.highpass('n,fst,ast',5,20,55,50)
d =
                       Response: 'Highpass'
                  Specification: 'N,Fst,Ast'
                    Description: {3x1 cell}
        NormalizedFrequency: false
```

```
                              Fs: 50
                    FilterOrder: 5
                           Fstop: 20
                           Astop: 55
>> hd=design(d,'cheby2')
hd =
                FilterStructure: 'Direct-Form II, Second-Order Sections'
                      Arithmetic: 'double'
                       sosMatrix: [3x6 double]
                     ScaleValues: [4x1 double]
          OptimizeScaleValues: true
              PersistentMemory: false

>> fvtool(hd)              %利用 fvtool 函数设计滤波器频率特性，效果如图 5-24 所示
```

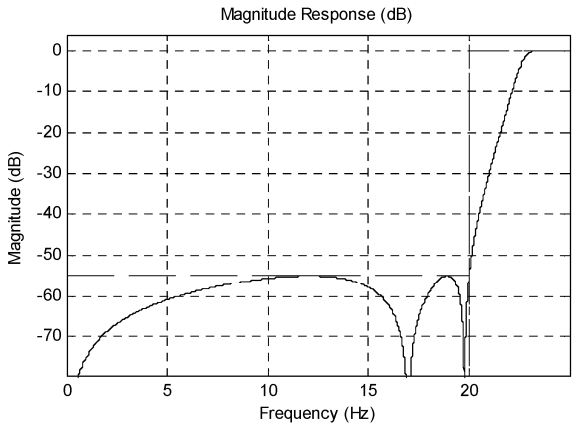


图 5-24　cheby2 函数设计的滤波器频率特性效果图

【例 5-23】设计一个高通椭圆滤波器，设计性能指标为：通带边界频率 wp=1500Hz，阻带边界频率 ws=1000Hz，通带波纹 Rp=1dB，阻带衰减 Rs=30dB。假设一个信号 x(t)=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t)，其中 f1=400Hz，f2=1600Hz。信号的采样频率为 10000Hz。试将原信号与通过该滤波器的模拟信号进行比较。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
% 滤波器的参数设计
wp=1500*2*pi; ws=1000*2*pi;
Rp=1;   Rs=30;
%求滤波器的最小阶数和截止频率
[n,wn]=ellipord(wp,ws,Rp,Rs,'s');
[b,a]=ellip(n,Rp,Rs,wn,'high','s');          %设计高通椭圆滤波器
w=linspace(1,3000,1000)*2*pi;                %给出计算复数频率响应的频率点
h=freqs(b,a,w);                              %计算给定频率的复数频率响应
magH=abs(h);                                 %求幅频响应
phaH=unwrap(angle(h));                       %求相频响应
figure; plot(w/(2*pi),2*log10(magH));        %以频率为横轴绘幅频响应
```

```
xlabel('频率/Hz');ylabel('振幅/dB');
title('椭圆模拟高通滤波器');
hold on;
plot([1000 1000],ylim,'r');                    %阻带边界
grid on;
figure; dt=1/10000;                            %采样间隔
f1=400; f2=1600;                               %信号中所含频率成分
t=0:dt:0.04;                                   %时间序列
x=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t);           %输入信号
H=[tf(b,a)];                                   %滤波器在 MATLAB 系统中的表示
[y,t1]=lsim(H,x,t);                            %模拟输出
subplot(2,1,1);plot(t,x);                      %绘制输入信号
title('输入信号');xlabel('时间/s');
subplot(2,1,2);plot(t1,y);                     %绘制输入信号
title('输出信号');xlabel('时间/s');
```

运行程序，效果如图 5-25 及图 5-26 所示。可见所设计的高通滤波器完全符合要求，在阻带边界下降的分贝数达 30dB，并且通带和阻带均有波纹，这正是椭圆滤波器的特点。将该滤波器模拟输入 400Hz 和 1600Hz 的信号后，输出结果完全滤除了 400Hz 低频成分，只含有 1600Hz 的信号。

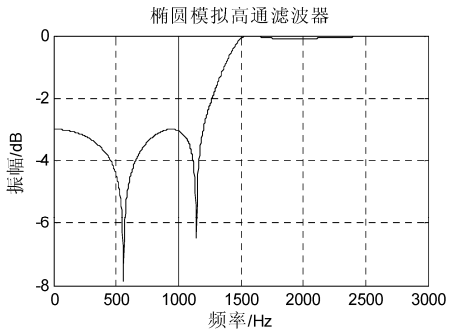


图 5-25　设计滤波器的幅频响应图

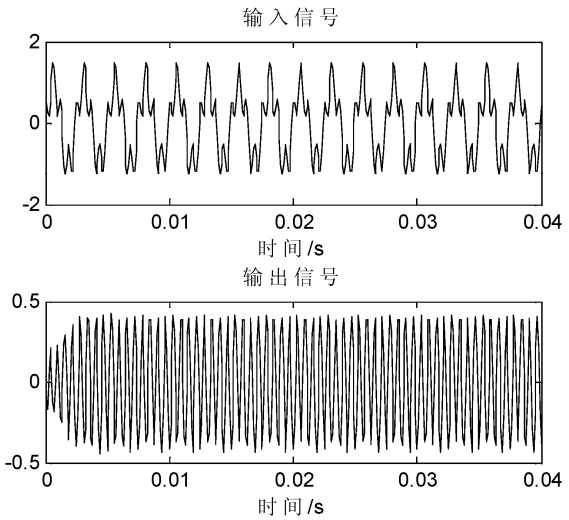


图 5-26　滤波器的输入和输出信号

【例 5-24】设计一个五阶贝塞尔低通模拟滤波器，截止频率为 1000rad/s，绘制滤波器的频率特性图。假设一个信号 x(t)=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t)，其中 f1=100Hz，f2=1000Hz。信号的采样频率为 10000Hz。试将原信号与通过该滤波器的模拟信号进行比较。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=5;                                    %滤波器阶数
wn=1000;                                %边界频率
[b,a]=besself(N,wn);                    %设计 Bessel 滤波器
figure;
[h,w]=freqs(b,a,512);                   %绘出滤波器的复数频率特性
magH=abs(h);                            %求幅频响应
phaH=unwrap(angle(h));                  %求相频响应
figure;
subplot(2,1,1);plot(w/(2*pi),2*log10(magH)); %以频率为横轴绘幅频响应
xlabel('频率/Hz');ylabel('振幅/dB');
grid on;
subplot(2,1,2);plot(w/2/pi,angle(h)*180/pi);  %绘制相频响应
grid on;
xlabel('频率/Hz');ylabel('相位/^o');
figure; dt=1/10000;                     %采样间隔
f1=100; f2=1000;                        %信号中所含频率成分
t=0:dt:0.1;                             %时间序列
x=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t);    %输入信号
H=[tf(b,a)];                            %滤波器在 MATLAB 系统中的表示
[y,t1]=lsim(H,x,t);                     %模拟输出
subplot(2,1,1);plot(t,x);               %绘制输入信号
title('输入信号');xlabel('时间/s');
subplot(2,1,2);plot(t1,y);              %绘制输入信号
title('输出信号');xlabel('时间/s');
```

运行程序，效果如图 5-27 及图 5-28 所示。虽然模拟贝塞尔滤波器的幅频特性不如其他的滤波器好，但其相频特性为线性的。因此图 5-28 中的输出信号与输入的低频信号形状不变，只是有延迟。

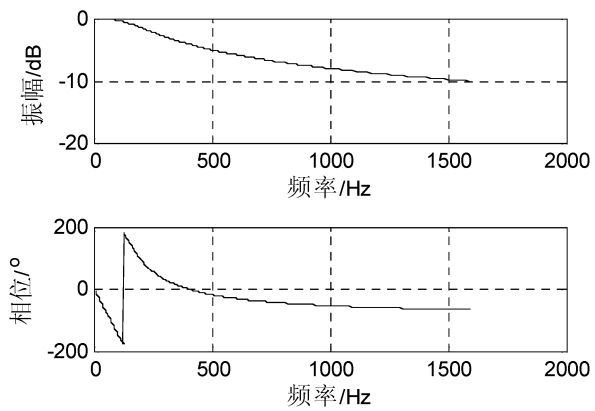


图 5-27　设计贝塞尔滤波器的频率特性

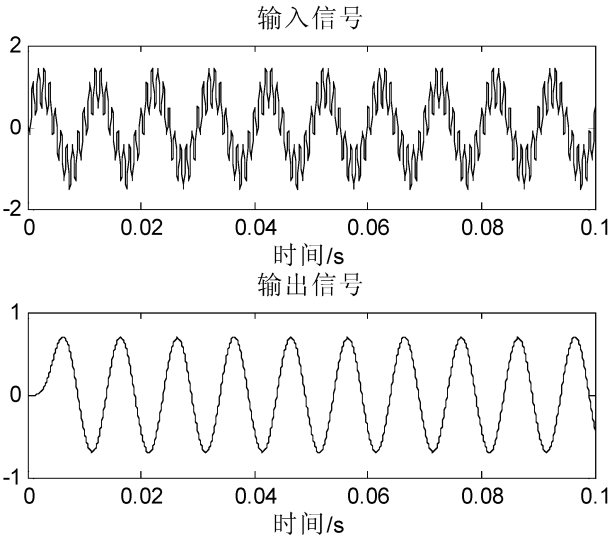


图 5-28　贝塞尔滤波器的输入和输出

第 6 章　IIR 滤波器设计

6.1　IIR 滤波器结构

一个 IIR 滤波器的系统函数为

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{m=0}^{M}b_m z^{-m}}{\sum_{n=0}^{N}a_n z^{-n}}=\frac{b_0+b_1z^{-1}+\cdots+b_M z^{-M}}{1+a_1z^{-1}+\cdots+a_N z^{-N}},a_0=1 \qquad (6\text{-}1)$$

式中：b_m，a_n 是滤波器的系数。不失一般性，假设 $a_0=1$。如果 $a_N\neq0$，这时 IIR 滤波器阶数为 N。IIR 滤波器的差分方程为

$$y(n)=\sum_{m=0}^{M}b_m x(n-m)-\sum_{n=0}^{N}a_n y(n-m) \qquad (6\text{-}2)$$

在工程实际中通过 3 种结构来实现 IIR 滤波器：直接形式、级联形式和并联形式，下面分别对它们加以说明。

6.1.1　直接型

1．直接Ⅰ型

设系统输入/输出关系的 N 阶差分方程为

$$y(n)=\sum_{k=0}^{N}a_k y(n-k)+\sum_{k=0}^{M}b_k x(n-k) \qquad (6\text{-}3)$$

这就是一种由差分方程直接实现的方式，又称为直接Ⅰ型结构。

结构的特点如下：

（1）两个网络级联：第一个横向结构 M 节延时网络实现零点，第二个有反馈的 N 节点延时网络实现极点。

（2）共需（N+M）级延时单元。

（3）缺点是系数不是直接决定单个零极点，因而不能很好地进行滤波器性能控制；极点对系数的变化过于灵敏，从而使系统频率响应对系数变化过于灵敏，也就是对有限精度（有限字长）运算过于灵敏，容易出现不稳定或产生较大误差。

2．直接Ⅱ型

一个线性移不变系统，若变换其级联子系统的次序，系统函数不变。把该原理应用于直接 I 型结构，即：

（1）交换两个级联网络的次序。

（2）合并两个具有相同输入的延时支路。

这样就得到另一种结构，称为直接Ⅱ型结构。此结构特点如下：

（1）N 节延时网络实现极点，第二个横向结构 M 节延时网络实现零点。

（2）实现 N 阶滤波器（一般 N≥M）只需 N 级延时单元，所需延时单元最少，故又称典范型。

（3）同直接Ⅰ型一样，具有直接型实现的一般缺点。

在 MATLAB 中，提供了 filter 函数实现 IIR 的直接形式。其调用格式如下：

y = filter(b,a,X)

其中：b 表示系统传递函数的分子多项式的系数矩阵；a 表示系统传递函数的分母多项式的系数矩阵；x 表示输入序列；y 表示输出序列。

【例 6-1】filter 用法示例。

```
>> data = [1:0.2:4]';
windowSize = 5;
filter(ones(1,windowSize)/windowSize,1,data)
```

运行程序，输出如下：

```
ans =
    0.2000
    0.4400
    0.7200
    1.0400
    1.4000
    1.6000
    1.8000
    2.0000
    2.2000
    2.4000
    2.6000
    2.8000
    3.0000
    3.2000
    3.4000
    3.6000
```

【例 6-2】用直接型实现系统函数为

$$H(z)=\frac{1-3z^{-1}+11z^{-2}+27z^{-3}+18z^{-4}}{1+16z^{-1}+12z^{-2}+2z^{-3}-18z^{-4}-z^{-5}}$$

的 IIR 数字滤波器，求单位脉冲响应和单位阶跃信号的输出。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
b=[1 -3 11 27 18];
a=[16 12 2 -4 -1];
N=25;
h=impz(b,a,N);                  %直接型单位脉冲响应
x=[ones(1,5),zeros(1,N-5)];     %单位阶跃信号
y=filter(b,a,x);                %直接型输出信号
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
```

```
xlabel('(b)  直接型 y(n)');
```

运行程序，效果如图 6-1 所示。

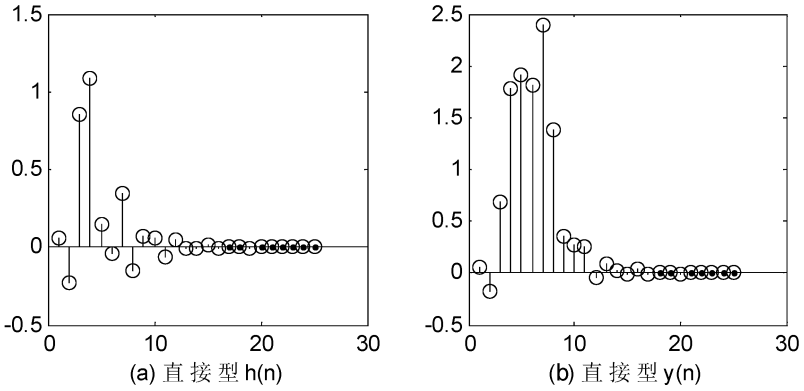


图 6-1　直接型单位脉冲响应和输出信号

6.1.2　级联结构与并联结构

1．级联结构

将系统函数按零、极点进行因式分解，表示为基本二阶子系统的乘积形式：

$$H(z)=\frac{\sum_{k=0}^{M}b_k z^{-k}}{1-\sum_{k=1}^{N}a_k z^{-k}}=\frac{Y(z)}{X(z)} \tag{6-4}$$

$$H(z)=A\prod_k\frac{1+\beta_{1k}z^{-1}+\beta_{2k}z^{-2}}{1+\alpha_{1k}z^{-1}-\alpha_{2k}z^{-2}}=A\prod_k H_k(z)$$

它的实现结构即可表示为基本二阶节 $H_k(z)$ 的级联形式。每个二阶节用直接 II 型实现。

级联结构的特点：

（1）每个二阶节系数单独控制一对零点和一对极点，有利于控制频率响应。

（2）分子分母中二阶因子配合成基本二阶节的方式以及各二阶节的排列次序不同，其级联结构也不同，它们表示同一个 $H(z)$，但有限精度运算时带来的误差是不同的。

在 MATLAB 中，给定直接型系统结构的系数，可以计算出相应级联结构的各项系数。

1）级联型系统结构的实现

```
function y=casfilter(b0,B,A,x)
% IIR 和 FIR 滤波器级联型的实现
% y=casfilter(b0,B,A,x)
% y 为输出序列
% b0 为级联型的增益系数
% B 为包含各 bk 的 k 乘三维实系数矩阵
% A 为包含各 ak 的 k 乘三维实系数矩阵
% x 为输入序列
[k,v]=size(B);
n=length(x);
w=zeros(k+1,n);
```

```
w(1,:)=x;
for i=1:k,
    w(i+1,:)=filter(B(i,:),A(i,:),w(i,:));
end
y=b0*w(k+1,:);
```

【例 6-3】用级联结构实现系统函数：

$$H(z)=\frac{4(1+z^{-1})(1-1.4142136z^{-1}+z^{-2})}{(1-0.5z^{-1})(1+0.9z^{-1}+0.81z^{-2})}$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
b0=4; N=60;
B=[1 1 0;1 -1.4142136 1]; A=[1 -0.5 0;1 0.9 0.81];
delta=impseq(0,0,N);
h=casfilter(b0,B,A,delta);
x=[ones(1,5),zeros(1,N-5)];
y=casfilter(b0,B,A,x);
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，输出效果如图 6-2 所示。

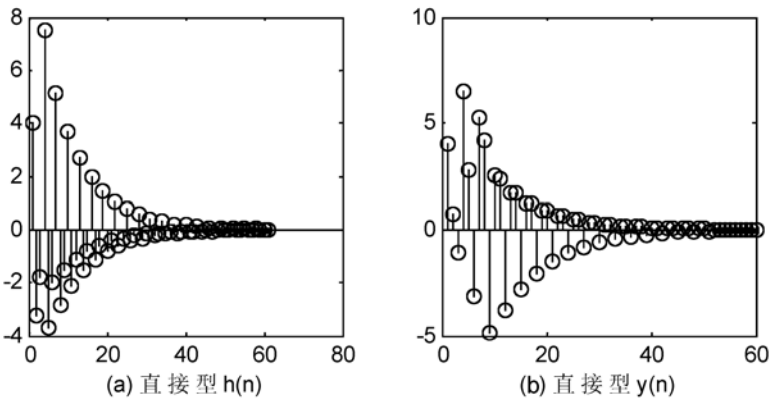


图 6-2　级联型单位脉冲响应和输出信号

在运行程序过程中，调用到用户自定义编写的单位采样序列、单位阶跃序列的函数。其源代码如下：

```
function [x,n]=impseq(n0,n1,n2)
%Generate x(n)=delta(n-n0);n1<=n<=n2;
n=[n1:n2];
x=[(n-n0)==0];
```

2）直接型系统结构转换为级联型系统结构

```
function [b0,B,A]=dir2cas(b,a)
%直接型到级联型形式的转换
% [b0,B,A]=dir2cas(b,a)
% a 为直接型的分子多项式系数
```

```
% b 为直接型的分母多项式系数
% b0 为增益系数
% B 为包含各 bk 的 k 乘二维实系数矩阵
% A 为包含各 ak 的 k 乘三维实系数矩阵
%计算增益系数
b0=b(1);
b=b/b0;
a0=a(1);
a=a/a0;
b0=b0/a0;
M=length(b);
N=length(a);
if N>M,
    b=[b,zeros(1,N-M)];
elseif, M>N
a=[a,zeros(1,M-N)];
    N=M;
else
    NM=0;
end
k=floor(N/2);
B=zeros(k,3);
A=zeros(k,3);
if k*2==N
    b=[b,0];
    a=[a,0];
end
broots=cplxpair(roots(b));
aroots=cplxpair(roots(a));
for i=1:2:2*k,
    br=broots(i:1:i+1,:);
    br=real(poly(br));
    B(fix((i+1)/2),:)=br;
    ar=aroots(i:1:i+1,:);
    ar=real(poly(ar));
    A(fix((i+1)/2),:)=ar;
end
```

【例 6-4】用级联实现系统函数为

$$H(z)=\frac{1-3z^{-1}+11z^{-2}+27z^{-3}+18z^{-4}}{1+16z^{-1}+12z^{-2}+2z^{-3}-18z^{-4}-z^{-5}}$$

的 IIR 数字滤波器，求单位脉冲响应和单位阶跃信号的输出。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
b=[1 -3 11 27 18];
a=[16 12 2 -4 -1];
N=25;
```

```
delta=impseq(0,0,N);
[b0,B,A]=dir2cas(b,a);                %直接型转换为级联型
h=casfilter(b0,B,A,delta);            %级联型单位脉冲响应
x=[ones(1,5),zeros(1,N-5)];   %单位阶跃信号
y=casfilter(b0,B,A,x);                %级联型输出信号
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，输出效果如图 6-3 所示。

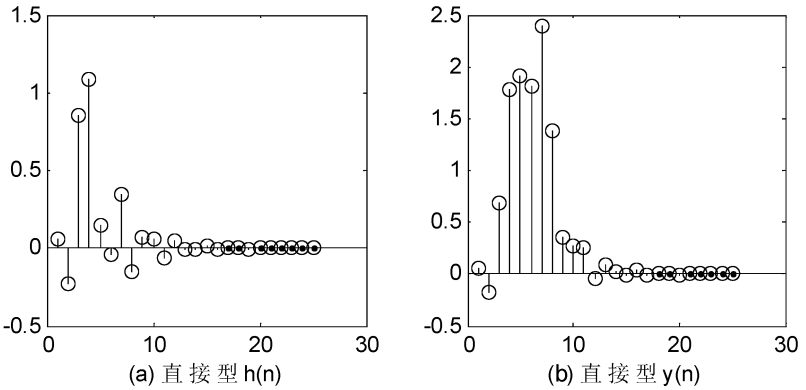


图 6-3　级联型单位脉冲响应和输出信号

3）级联型结构转换为直接型结构

```
function [b,a]=cas2dir(b0,B,A)
% 级联型到直接型的形式转换
% b 为直接型分子多项式系数
% a 为直接型分母多项式系数
% b0 为增益系数
% B 为包含各 bk 的 k 乘二维系数矩阵
% A 为包含各 ak 的 k 乘三维系数矩阵
[k,L]=size(B);
b=[1];
a=[1];
for i=1:1:k,
    b=conv(b,B(i,:));
    a=conv(a,A(i,:));
end
b=b*b0;
```

【例 6-5】用直接型结构实现系统函数为

$$H(z)=\frac{4(1+z^{-1})(1-1.4142136z^{-1}+z^{-2})}{(1-0.5z^{-1})(1+0.9z^{-1}+0.81z^{-2})}$$

的 IIR 数字滤波器，求单位脉冲响应和单位阶跃信号的输出。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
b0=4; N=60;
B=[1 1 0;1 -1.4142136 1]; A=[1 -0.5 0;1 0.9 0.81];
delta=impseq(0,0,N);
x=[ones(1,5),zeros(1,N-1)];
[b,a]=cas2dir(b0,B,A);
h=filter(b,a,delta);     %直接型单位脉冲响应
y=filter(b,a,x);     %直接型输出信号
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，效果如图 6-4 所示。

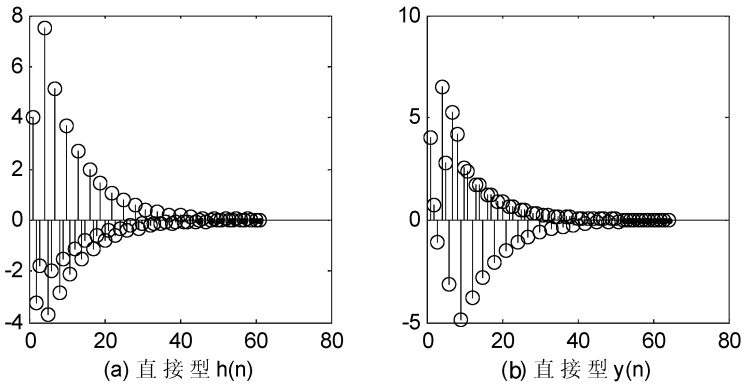


图 6-4　直接型单位脉冲响应和输出信号

2．并联结构

将因式分解的 $H(z)$ 展成部分分式的形式，表示为二阶子系统的和形式：

$$H(z)=G_0+\sum_k \frac{\gamma_{0k}+\gamma_{1k}z^{-1}}{1-\alpha_{1k}z^{-1}-\alpha_{2k}z^{-2}}=G_0\sum_k H_k(z) \qquad (6\text{-}5)$$

其实现结构表示为基本二阶节的并联形式。当 M<N 时，$G_0=0$。

并联结构的特点：

（1）能精确调整每对极点，但全体零点随任一并联二阶节系数变化而变化，因此不适用于要求精确传输零点的场合。

（2）各节误差相互无影响，累加即可，故比级联型的误差一般来说稍小一些。

1）并联型系统结构的实现

```
function y=parfilter(C,B,A,x)
% IIR 滤波器的并联实现
% [y]=parfilter(C,B,A,x)
% y 为输出序列
% C 为当 b 的长度等于 a 时的多项式部分
% B 为包含各 bk 的 K 乘二维实系数矩阵
```

```
% A 为包含各 ak 的 K 乘三维实系数矩阵
% x 为输入序列
[K,L]=size(B);
N=length(x);
w=zeros(K+1,N);
w(1,:)=filter(C,1,x);
for i=1:1:K,
      w(i+1,:)=filter(B(i,:),A(i,:),x);
end
y=sum(w);
```

【例 6-6】用并联型实现系统函数为

$$H(z)=\frac{-14.75-12.90z^{-1}}{1-\frac{7}{8}z^{-1}+\frac{3}{32}z^{-2}}+\frac{24.50+26.82z^{-1}}{1-z^{-1}+\frac{1}{2}z^{-2}}$$

的滤波器。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
C=0;B=[-14.75 -12.90;24.50 26.82];
A=[1,-7/8,3/32;1,-1,0.5];N=60;
delta=impseq(0,0,N);
h=parfilter(C,B,A,delta);
x=[ones(1,5),zeros(1,N-5)];
y=parfilter(C,B,A,x);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，效果如图 6-5 所示。

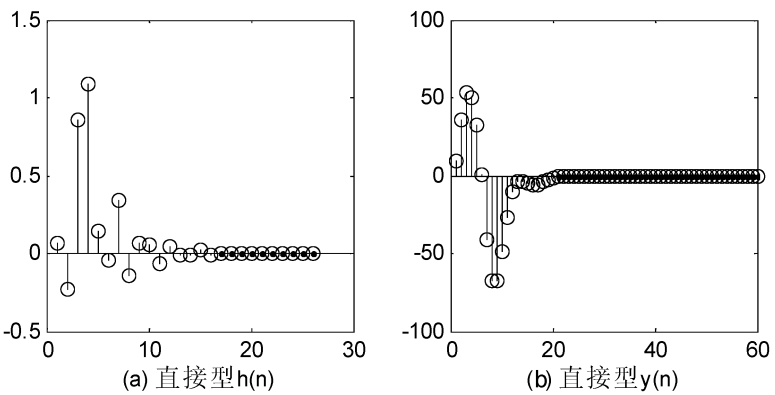


图 6-5　并联型单位脉冲响应和输出信号

2）直接型系统结构转换为并联型系统结构

```
function [C,B,A]=dir2par(b,a)
%  直接型到并联型的转换
%[C,B,A]=dir2par(b,a)
% C 为当 b 的长度大于 a 时的多项式部分
```

```
% B 为包含各 bk 和 K 乘二维实系数矩阵
% A 为包含各 ak 和 K 乘三维实系数矩阵
% b 为直接型分子多项式系数
% a 为直接型分母多项式系数
M=length(b);
N=length(a);
[r1,p1,C]=residuez(b,a);
p=cplxpair(p1,10000000*eps);
x=cplxcomp(p1,p);
r=r1(x);
K=floor(N/2);
B=zeros(K,2);
A=zeros(K,3);
if K*2==N,
    for i=1:2:N-2,
        br=r(i:1:i+1,:);
        ar=p(i:1:i+1,:);
        [br,ar]=residuez(br,ar,[]);
        B(fix((i+1)/2),:)=real(br');
        A(fix((i+1)/2),:)=real(ar');
    end
    [br,ar]=residuez(r(N-1),p(N-1),[]);
    B(K,:)=[real(br') 0];
    A(K,:)=[real(ar') 0];
else
    for i=1:2:N-1,
        br=r(i:1:i+1,:);
        ar=p(i:1:i+1,:);
        [br,ar]=residuez(br,ar,[]);
        B(fix((i+1)/2),:)=real(br);
        A(fix((i+1)/2),:)=real(ar);
    end
end
```

【例 6-7】用并联型实现系统函数为

$$H(z)=\frac{1-3z^{-1}+11z^{-2}+27z^{-3}+18z^{-4}}{1+16z^{-1}+12z^{-2}+2z^{-3}-18z^{-4}-z^{-5}}$$

的 IIR 数字滤波器，求单位脉冲响应和单位阶跃信号的输出。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
b=[1 -3 11 27 18];
a=[16 12 2 -4 -1];
N=25;
delta=impseq(0,0,N);
[C,B,A]=dir2par(b,a);
h=parfilter(C,B,A,delta);
x=[ones(1,5),zeros(1,N-5)];   %单位阶跃信号
```

```
y=casfilter(C,B,A,x);
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，效果如图 6-6 所示。

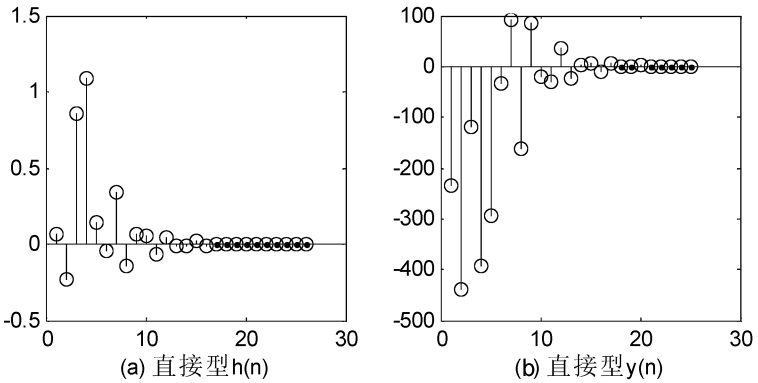


图 6-6　并联型单位脉冲响应和输出信号

在运行程序过程中，调用用户自定义编写的 cplxcomp 函数，把两个混乱的复数数组进行比较，返回一个数组的下标，用它重新给另一个数组排序。其源代码如下：

```
function I=cplxcomp(p1,p2)
% I=cplxcomp(p1,p2)
%  比较两个包含同样标量元素但（可能）有不同下标的复数对
%  本程序必须用在 cplxpair()程序之后以便重新排序频率极点矢量
%  及其相应的留数矢量
% p2=cplxpair(p1)
I=[];
for i=1:length(p2)
      for j=1:length(p1)
            if(abs(p1(j)-p2(i))<0.0001)
                  I=[I,j];
            end
      end
end
I=I';
```

3）并联型系统结构转换为直接型系统结构

```
function [b,a]=par2dir(C,B,A)
%并联模型到直接型的转换
% [b,a]=par2dir(C,B,A)
% C 为当 b 的长度大于 a 时的多项式部分
% B 为包含各 bk 的 K 乘二维实系数矩阵
% A 为包含各 ak 的 K 乘三维实系数矩阵
% b 为直接型分子多项式系数
% a 为直接型分母多项式系数
[K,L]=size(A);
```

```
R=[];
P=[];
for i=1:1:K
    [r,p,k]=residuez(B(i,:),A(i,:));
    R=[R;r];
    P=[P;p];
end
[b,a]=residuez(R,P,C);
b=b(:)';
a=a(:)';
```

【例 6-8】用直接型实现系统函数为

$$H(z)=\frac{-14.75-12.90z^{-1}}{1-\frac{7}{8}z^{-1}+\frac{3}{32}z^{-2}}+\frac{24.50+26.82z^{-1}}{1-z^{-1}+\frac{1}{2}z^{-2}}$$

的滤波器。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
C=0;B=[-14.75 -12.90;24.50 26.82];
A=[1,-7/8,3/32;1,-1,0.5];N=60;
delta=impseq(0,0,N);
[b,a]=par2dir(C,B,A);
h=filter(b,a,delta);
x=[ones(1,5),zeros(1,N-5)];
y=filter(b,a,x);
subplot(1,2,1);stem(h);
xlabel('(a)  直接型 h(n)');
subplot(1,2,2);stem(y);
xlabel('(b)  直接型 y(n)');
```

运行程序，效果如图 6-7 所示。

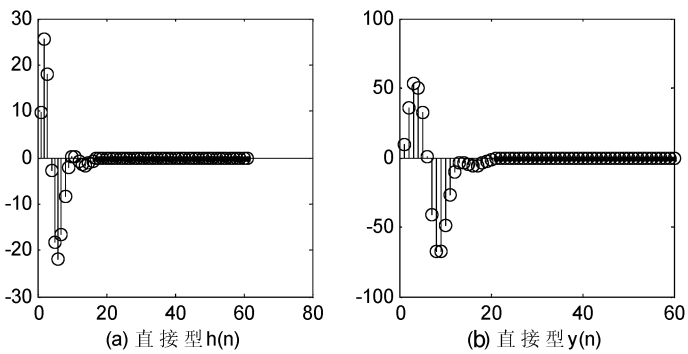


图 6-7　直接型单位脉冲响应和输出信号

6.2 常用模拟低通滤波器特性

6.2.1 振幅平方函数

为了方便学习数字滤波器，先讨论几种常用的模拟低通滤波器设计方法，高通、带通、带阻等模拟滤波器可利用变量方法，由低通滤波器变换得到。

模拟滤波器的设计就是根据一组设计规范来设计模拟系统函数 $H_a(s)$，使其逼近某个理想滤波器特性。

考虑因果系统

$$H_a(\mathrm{j}\Omega)=\int_0^{\infty}h_a(t)\mathrm{e}^{-\mathrm{j}\Omega t} \tag{6-6}$$

式中：$h_a(t)$ 为系统的单位冲激响应，是实函数。

所以有

$$H_a(\mathrm{j}\Omega)=\int_0^{\infty}h_a(t)(\cos\Omega t-\mathrm{j}\sin\Omega t) \tag{6-7}$$

不难看出

$$H_a(-\mathrm{j}\Omega)=H_a^*(\mathrm{j}\Omega) \tag{6-8}$$

定义振幅平方函数

$$\begin{aligned}A(\Omega^2)&=\left|H_a(\mathrm{j}\Omega)\right|^2=H_a(\mathrm{j}\Omega)H_a^*(\mathrm{j}\Omega)\\A(\Omega^2)&=H_a(\mathrm{j}\Omega)H_a(-\mathrm{j}\Omega)=H_a(s)H_a(-s)\big|_{s=\mathrm{j}\Omega}\end{aligned} \tag{6-9}$$

式中：$H_a(s)$、$H_a(\mathrm{j}\Omega)$、$\left|H_a(\mathrm{j}\Omega)\right|$ 分别为模拟滤波器的系统函数、频率响应和幅频特性。

6.2.2 模拟滤波器原型

IIR 滤波器设计技术依靠现有的模拟滤波器得到数字滤波器，工程实际当中把这些模拟滤波器叫做滤波器原型。在工程实际中应用最为广泛的有两种模拟滤波器，下面分别介绍。

1．巴特沃思低通滤波器

巴特沃思低通滤波器的主要特征是通带和阻带都有平坦的幅度响应。N 阶低通滤波器的平方幅度响应为

$$\left|H_a(\mathrm{j}\Omega)\right|=\frac{1}{1+\left(\dfrac{\Omega}{\Omega_c}\right)^N} \tag{6-10}$$

式中：N 为滤波器阶数；Ω_c 为截止频率（rad/s）。

MATLAB 提供了 buttap 函数来设计 N 阶归一化（即 Ω_c=1）巴特沃思模拟原型滤波器。其调用格式如下：

[z, p, k]=buttap(N)

其中：z、p、k 为返回的零点、极点及增益；N 为阶数。但在实际中往往需要具有任意 Ω_c 未归一化的巴特沃思滤波器，这就需要重新编写一个更符合工程实际应用的函数。

（1）编写一个求取未归一化的巴特沃思模拟低通滤波器函数。其源代码如下：

```
function [b,a]=butt_o(n,oga)
%未归一化的巴特沃思模拟低通滤波器原型
% b 为 Ha(s)的分子多项式系数
% a 为 Ha(s)的分母多项式系数
% n 为滤波器的阶数
% oga 为单位弧度/秒的截止频率
[z,p,k]=buttap(n);
p=p*oga;
k=k*oga^n;
B=real(poly(z));
b0=k;
b=k*B;
a=real(poly(p));
```

（2）编写一个适合于模拟滤波器的、将滤波器由直接形式转换为级联形式的函数。其源代码如下：

```
function [C,B,A]=sdir2cas(b,a)
%  平面中直接形式到级联形式的转换
% C 为增益系数,B 为包含各 bk 的 k 乘三维实系数矩阵,A 为包含各 ak 的 k 乘三维实系数矩阵,
% b 为直接形式的分子多项式系数,a 为直接形式的分母多项式系数
Na=length(a)-1;
Nb=length(b)-1;
b0=b(1);b=b/b0;
a0=a(1);a=a/a0;
C=b0/a0;
p=cplxpair(roots(a));K=floor(Na/2);
if K*2==Na
    A=zeros(K,3);
    for n=1:2:Na
        Arow=p(n:1:n+1,:);
        Arow=poly(Arow);
        A(fix((n+1)/2),:)=real(Arow);
    end
elseif Na==1
    A=[0 real(poly(p))];
else
    A=zeros(K+1,3);
    for n=1:2:2*K
        Arow=p(n:1:n+1,:);Arow=poly(Arow);
        A(fix((n+1)/2),:)=real(Arow);
    end
    A(K+1,:)=[0 real(poly(p(Na)))];
end
```

```
z=cplxpair(roots(b));K=floor(Nb/2);
if Nb==0
    B=[0 0 poly(z)];
elseif K*2==Nb
    B=zeros(K,3);
    for n=1:2:Nb
        Brow=z(n:1:n+1,:);Brow=poly(Brow);
        B(fix((n+1)/2),:)=real(Brow);
    end
elseif Nb==1
    B=[0 real(poly(z))];
else
    B=zeros(K+1,3);
    for n=1:2:2*K
        Brow=z(n:1:n+1,:);Brow=poly(Brow);
        B(fix((n+1)/2),:)=real(Brow);
    end
    B(K+1,:)=[0 real(poly(z(Nb)))];
end
```

（3）编写根据指定指标设计模拟巴特沃思低通滤波器的函数。其源代码如下：

```
function [b,a]=afd_butt(wp,ws,Rp,Rs)
% b 为 Ha(s)的分子多项式系数,a 为 Ha(s)的分母多项式系数,wp 为以弧度/秒为单位
%描述的通带边缘频率;wp>0,ws 为以弧度/秒为单位描述的阻带边缘频率;ws>wp>0,
% Rp 为通带中的振幅波动 dB 数,Rs 为阻带衰减的 dB 数
if wp<=0
    error('通带边缘必须大于 0')
end
if ws<=wp
    error('阻带边缘必须大于通带边缘')
end
if (Rp<=0)|(Rs<0)
    error('通带波动或阻带衰减必须大于 0')
end
N=ceil((log10((10^(Rp/10)-1)/(10^(Rs/10)-1)))/(2*log10(wp/ws)));
fprintf('\n ***Butterworth Filter Order=%2.0f\n',N);
OmegaC=wp/((10^(Rp/10)-1)^(1/(2*N)));
[b,a]=u_buttap(N,OmegaC)
```

（4）编写一个显示模拟滤波器频域特性的函数。其源代码如下：

```
function [db,mag,pha,w]=freqs_m(b,a,wmax)
% db 为[0,wmax]区间内的相对幅度
% mag 为[0,wmax]区间内的绝对幅度
% pha 为[0,wmax]区间内的相位响应
% w 为含[0,wmax]区间内 500 个频率采样点的数组
% b 为 Ha(s)的分子多项式系数
% a 为 Ha(s)的分母多项式系数
w1=0:500;
w=w1*wmax/500;
```

```
h=freqs(b,a,w);
mag=abs(h);
db=20*log10((mag+eps)/max(mag));
pha=angle(h);
```

（5）编写根据指定指标设计模拟巴特沃思低通滤波器函数。其源代码如下：

```
function [b,a]=a_butt(wp,ws,rp,as)
% 巴特沃思模拟低通滤波器的设计
% b 为 Ha(s)的分子多项式系数
% a 为 Ha(s)的分母多项式系数
% wp 为以弧度/秒为单位描述的通带边缘频率：wp>0
% ws 为以弧度/秒为单位描述的阻带边缘频率：ws>wp>0
% rp 为通带中的振幅波动的 dB 数
% as 为阻带衰减的 dB 数
if wp<=0
    error('通带边缘必须大于 0');
end
if ws<=wp,
    error('阻带边缘必须大于通带边缘');
end
if (rp<=0)| (as<0),
    error('通带波动或阻带波动衰减必须大于 0');
end
n=ceil((log10((10^(rp/10)-1)/(10^(as/10)-1)))/(2*log10(wp/ws)));
fprintf('\n***Butterworth Filter Order=%2.0f\n',n)
omega=wp/((10^(rp/10)-1)^(1/(2*n)));
[b,a]=butt_o(n,omega);
```

（6）编写非归一化巴特沃思模拟低通滤波器原型。其源代码如下：

```
function [b,a]=u_buttap(N,OmegaC)
[z,p,k]=buttap(N);
p=p*OmegaC;
k=k*OmegaC^N;
B=real(poly(z));
b0=k;
b=k*B;
a=real(poly(p));
```

【例 6-9】巴特沃思低通滤波器的设计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%参数设计如下：
wp=0.2*pi; ws=0.3*pi;
Rp=7; Rs=16;
Ripple=10^(-Rp/20); Attn=10^(-Rs/20);
%模拟滤波器设计
[b,a]=afd_butt(wp,ws,Rp,Rs);
%设计二阶环节
[C,B,A]=sdir2cas(b,a)
%计算频率响应
```

```
[db,mag,pha,w]=freqs_m(b,a,0.5*pi);
%计算脉冲响应
[ha,t]=impz(b,a);
%画图
subplot(2,2,1);plot(w/pi,mag);
title('幅度响应');axis([0 0.5 0 1.1]);
xlabel('');ylabel('|H|');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[0 Attn Ripple 1]);
grid on;
subplot(2,2,2);plot(w/pi,db);
title('幅度(dB)');axis([0 0.5 -30 5]);
xlabel('');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-30 -Rs -Rp 0]);
grid on;
subplot(2,2,3);plot(w/pi,pha/pi);
title('相位响应');axis([0 0.5 -1 1]);
xlabel('模拟频率(单位:pi)');ylabel('弧度');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-1 -0.5 0 0.5 1]);
grid on;
subplot(2,2,4);plot(t,ha,[0,max(t)],[0,0]);
title('脉冲响应');axis([[0 max(t) min(ha) max(ha)]]);
xlabel('时间/s');ylabel('ha(t)');
grid on;
```

运行程序，输出如下，效果如图 6-8 所示。

```
 ***Butterworth Filter Order= 3
b =
    0.1238
a =
    1.0000    0.9969    0.4969    0.1238
C =
    0.1238
B =
     0     0     1
A =
    1.0000    0.4985    0.2485
         0    1.0000    0.4985
```

2．切比雪夫低通滤波器

切比雪夫滤波器有两种，切比雪夫Ⅰ型滤波器在能带中具有等波动响应，而切比雪夫Ⅱ型滤波器在阻带具有等波动响应。

切比雪夫Ⅱ型滤波器的平方幅度响应为

$$\left|H_a(\mathrm{j}\Omega)\right|^2=\frac{1}{1+\varepsilon^2 T_N^2\left(\dfrac{\Omega}{\Omega_c}\right)} \tag{6-11}$$

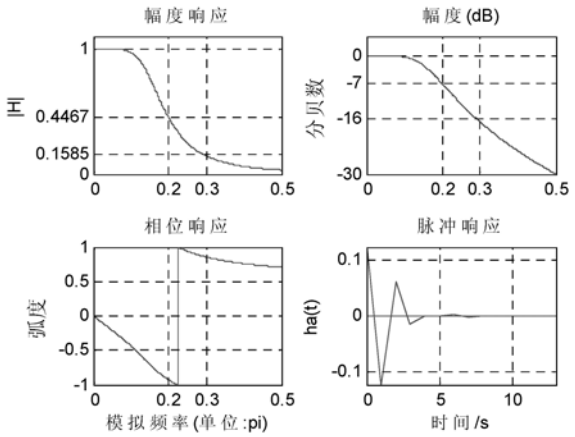


图 6-8　巴特沃思滤波器响应

式中：N 为滤波器的阶数；ε 为通带波动系数，它与 R_p 有关；$T_N(x)$ 是 N 阶切比雪夫多项式：

$$T_N(x)=\begin{cases}\cos(N\arccos(x)), & 0\leqslant x\leqslant 1\\ \cosh(\operatorname{arcosh}(x)), & 1<x<\infty\end{cases};\quad x=\frac{\Omega}{\Omega_c} \tag{6-12}$$

切比雪夫Ⅱ型滤波器的平方幅度响应如下所示：

$$\left|H_a(\mathrm{j}\Omega)\right|^2=\frac{1}{1+\varepsilon^2 T_N^2\left(\dfrac{\Omega}{\Omega_c}\right)^{-1}} \tag{6-13}$$

式（6-13）中各参数含义与式（6-11）相同。

MATLAB 提供了 cheb1ap 函数来设计 N 阶的Ⅰ型切比雪夫模拟原型滤波器，其调用格式如下：

[z, p, k]=cheb1ap(N, Rp)

其中：返回数组 z，p，k 为其零点、极点及增益；Rp 为通带波动；N 为阶数。但是在工程实际中往往需要具有任意 Ω_c 的未归一化的切比雪夫滤波器，这就需要重新编写符合工程实际应用的函数。

（1）编写一个求取未归一化的切比雪夫Ⅰ型模拟低通滤波器函数。其源代码如下：

```
function [b,a]=cheblap_o(n,rp,omega)
%未归一化的切比雪夫Ⅰ型模拟低通滤波器
% b 为 Ha(s)的分子多项式系数,a 为 Ha(s)分母多项式的系数
% n 为滤波器的阶数,rp 为通带的波动 dB 数,oemga 为以弧度/秒的单位截止频率
[z,p,k]=cheb1ap(n,rp);
a=real(poly(p));
```

```
ann=a(n+1);
p=p*omega;
a=real(poly(p));
anu=a(n+1);
k=k*anu/ann;
b0=k;
B=real(poly(z));
b=k*B;
```

（2）编写根据给定指标设计切比雪夫Ⅰ型模拟低通滤波器函数。其源代码如下：

```
function [b,a]=afd_cheb1(wp,ws,rp,as)
%切比雪夫Ⅰ型模拟低通滤波器的设计,b 为 Ha(s)分子多项式的系数
% a 为 Ha(s)分母多项式的系数,wp 为弧度/秒为单位的通带边缘频率：wp>0
% ws 为以弧度/秒为单位的阻带边缘频率：ws>wp>0
% rp 为通带中的振幅波动的 dB 数,as 为阻带衰减的 dB 数
if wp<=0
    error('通带边缘必须大于 0');
end
if ws<=wp,
    error('阻带边缘必须大于通带边缘');
end
if (rp<=0)| (as<0),
    error('通带波动或阻带波动衰减必须大于 0');
end
ep=sqrt(10^(rp/10)-1);
a=10^(as/20);
omega_c=wp;
omega_r=ws/wp;
g=sqrt(a*a-1)/ep;
n=ceil(log10(g+sqrt(g*g-1))/log10(omega_r+sqrt(omega_r*omega_r-1)));
fprintf('\n***切比雪夫Ⅰ型滤波器的阶次=%2.0f\n',n)
[b,a]=cheblap_o(n,rp,omega_c);
```

【例 6-10】切比雪夫Ⅰ型低通滤波器设计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%参数设计如下：
wp=0.2*pi; ws=0.3*pi;
Rp=1; Rs=16;
Ripple=10^(-Rp/20); Attn=10^(-Rs/20);
%模拟滤波器设计
[b,a]=afd_cheb1(wp,ws,Rp,Rs);
%设计二阶环节
[C,B,A]=sdir2cas(b,a)
%计算频率响应
[db,mag,pha,w]=freqs_m(b,a,0.5*pi);
%计算脉冲响应
[ha,t]=impz(b,a);
%画图
```

```
subplot(2,2,1);plot(w/pi,mag);
title('幅度响应');axis([0 0.5 0 1.1]);
xlabel('模拟频率(单位:pi)');ylabel('|H|');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[0 Attn Ripple 1]);
grid on;
subplot(2,2,2);plot(w/pi,db);
title('幅度(dB)');axis([0 0.5 -30 5]);
xlabel('');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-30 -Rs -Rp 0]);
grid on;
subplot(2,2,3);plot(w/pi,pha/pi);
title('相位响应');axis([0 0.5 -1 1]);
xlabel('模拟频率(单位:pi)');ylabel('弧度');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-1 -0.5 0 0.5 1]);
grid on;
subplot(2,2,4);plot(t,ha,[0,max(t)],[0,0]);
title('脉冲响应');axis([[0 max(t) min(ha) max(ha)]]);
xlabel('时间/s');ylabel('ha(t)');
grid on;
```

运行程序，输出如下，效果如图 6-9 所示。

```
***切比雪夫Ⅰ型滤波器的阶次= 4
C =
    0.0383
B =
     0     0     1
A =
    1.0000    0.4233    0.1103
    1.0000    0.1753    0.3895
```

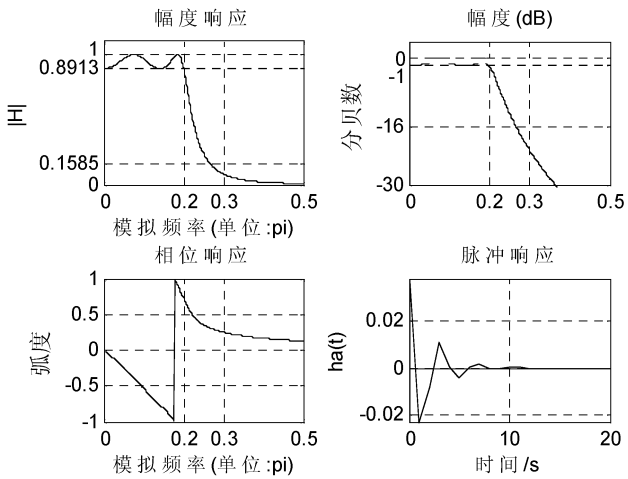


图 6-9　切比雪夫Ⅰ型滤波器响应

MATLAB 提供了 cheb2ap 函数来设计 N 阶的 II 型加比雪夫模拟原型滤波器，其调用格式如下：

[z, p, k]=cheb2ap(N, Rp)

其中：返回数组 z，p，k 为其零点、极点及增益；Rp 为通带波动；N 为阶数。但是在工程实际中往往需要具有任意 Ω_c 的未归一化的切比雪夫滤波器，这就需要重新编写符合工程实际应用的函数。

（1）编写一个求取未归一化的切比雪夫Ⅱ型模拟低通滤波器。其源代码如下：

```
function [b,a]=cheb2ap_o(n,Rs,Omega)
% 未归一化的切比雪夫Ⅱ型模拟低通滤波器
% b 为 Ha(s)的分子多项式系数,a 为 Ha(s)的分母多项式系数
% N 为滤波器的阶数,As 为阻带波动的 dB 数,Omega 为以弧度/秒为单位的截止频率
[z,p,k]=cheb2ap(n,Rs);
a=real(poly(p));
aNn=a(n+1);
p=p*Omega;
a=real(poly(p));
aNu=a(n+1);
b=real(poly(z));
M=length(b);
bNn=b(M);
z=z*Omega;
b=real(poly(z));
bNu=a(M);
k=k*(aNu*bNu)/(aNn*bNu);
b0=k;
b=k*b;
```

（2）编写根据给定指标设计切比雪夫Ⅱ型模拟低通滤波器的函数。其源代码如下：

```
function [b,a]=afd_cheb2(wp,ws,Rp,Rs)
% 切比雪夫Ⅱ型模拟低通滤波器设计
% b 为 Ha(s)的分子多项式系数,a 为 Ha(s)的分母多项式系数
% wp 为以弧度/s 为单位的通带边缘频率;wp>0,ws 为以弧度/秒为单位的阻带边缘频率;ws>wp>0
% Rp 为通带波动的 dB 数,Rs 为阻带衰减的 dB 数
if wp<=0
    error('通带必须大于 0');
end
if ws<=wp
    error('阻带边缘必须大于通带边缘');
end
if (Rp<=0) | (Rs<0)
    error('通带波动或阻带衰减必须大于 0');
end
ep=sqrt(10^(Rp/10)-1);
r=10^(Rs/20);
Omegar=wp;
Omegaw=ws/wp;
g=sqrt(r*r-1)/ep;
```

```
n=ceil(log10(g+sqrt(g*g-1))/log10(Omegaw+sqrt(Omegaw*Omegaw-1)));
fprintf('\n ***切比雪夫 II 型滤波器的阶次=%2.0f\n',n);
[b,a]=cheb2ap_o(n,Rs,Omegar);
```

【例 6-11】切比雪夫Ⅱ型低通滤波器设计。

```
>> clear all;
%参数设计如下
wp=0.2*pi; ws=0.3*pi;
Rp=1; Rs=16;
Ripple=10^(-Rp/20); Attn=10^(-Rs/20);
%模拟滤波器设计
[b,a]=afd_cheb2(wp,ws,Rp,Rs);
%设计二阶环节
[C,B,A]=sdir2cas(b,a)
%计算频率响应
[db,mag,pha,w]=freqs_m(b,a,0.5*pi);
%计算脉冲响应
[ha,t]=impz(b,a);
%画图
subplot(2,2,1);plot(w/pi,mag);
title('幅度响应');axis([0 0.5 0 1.1]);
xlabel('模拟频率(单位:pi)');ylabel('|H|');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[0 Attn Ripple 1]);
grid on;
subplot(2,2,2);plot(w/pi,db);
title('幅度(dB)');axis([0 0.5 -30 5]);
xlabel('');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-30 -Rs -Rp 0]);
grid on;
subplot(2,2,3);plot(w/pi,pha/pi);
title('相位响应');axis([0 0.5 -1 1]);
xlabel('模拟频率(单位:pi)');ylabel('弧度');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 0.5]);
set(gca,'YTickMode','manual','YTick',[-1 -0.5 0 0.5 1]);
grid on;
subplot(2,2,4);plot(t,ha,[0,max(t)],[0,0]);
title('脉冲响应');axis([[0 max(t) min(ha) max(ha)]]);
xlabel('时间/s');ylabel('ha(t)');
grid on;
```

运行程序，输出如下，效果如图 6-10 所示。

```
 ***切比雪夫Ⅱ型滤波器的阶次= 4
C =
    0.0247
B =
    1.0000         0    2.6958
    1.0000         0    0.4625
```

```
A =
    1.0000    1.3014    0.6554
    1.0000    0.2480    0.3015
```

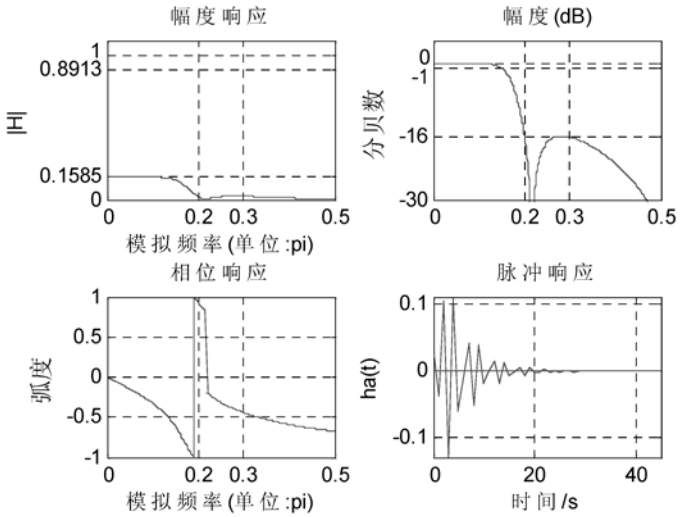


图 6-10　切比雪夫Ⅱ型滤波器响应

6.3　从模拟滤波器设计 IIR 滤波器

6.3.1　脉冲响应不变法

1．变换原理

冲激响应不变法是使数字滤波器的单位冲激响应序列 $h(n)$ 模仿模拟滤波器的单位冲激响应 $h_a(t)$，将模拟滤波器的单位冲激响应加以等间隔抽样，使 $h(n)$ 正好等于 $h_a(t)$ 的抽样值，即满足：

$$h(n)=h_a(nT) \tag{6-14}$$

式中：T 为抽样周期。

抽样序列的 z 变换与模拟信号的拉普拉斯变换之间的关系为

$$H(z)\big|_{z=\mathrm{e}^{sT}}=\frac{1}{T}\sum_{k=-\infty}^{\infty}H_a\left(s-\mathrm{j}\frac{2\pi}{T}k\right) \tag{6-15}$$

可以看出，冲激响应不变法将模拟滤波器的 s 平面变换到数字滤波器的 z 平面，从 s 到 z 的变换关系为 $z=\mathrm{e}^{sT}$。其映射关系如图 6-11 所示。

图 6-11 中，s 平面每一条宽度为 $2\pi/T$ 的横条都将重叠地映射到整个 z 平面上，而每一横条的左半边映射到 z 平面单位圆以内，右半边映射到单位圆以外，而 s 平面虚轴（$\mathrm{j}\Omega$ 轴）映射到单位圆上，虚轴上每一段长为 $2\pi/T$ 的线段都映射到 z 平面单位圆上一周。

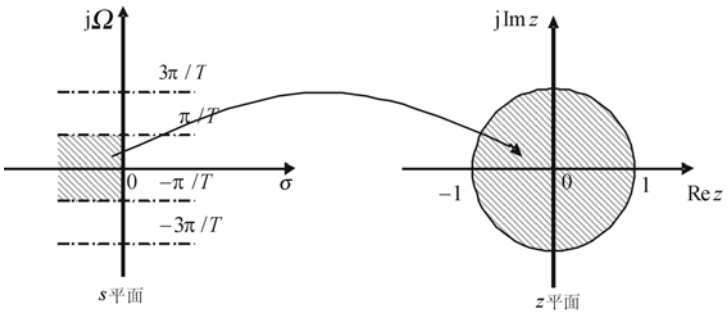


图 6-11　冲激响应不变法映射关系

由此可知，冲激响应不变法把稳定的 $H_a(s)$ 转换为稳定的 $H(z)$。由此方法可得到一阶系统的最基本的转换关系为

$$\frac{1}{s+a} \Rightarrow \frac{1}{1-\mathrm{e}^{-aTs}z^{-1}} \tag{6-16}$$

即 s 平面的单极点 $s=-a$ 映射到 z 平面的单极点 $z=\mathrm{e}^{-aTs}$。

2．混叠失真

由式（6-15）可知，数字滤波器的频率响应与模拟滤波器的频率响应间的关系为

$$H(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a\left(\mathrm{j}\frac{\Omega-2\pi}{T}k\right) \tag{6-17}$$

即数字滤波器的频率响应是模拟滤波器频率响应的周期延拓。根据奈奎斯特抽样定理,只有当模拟滤波器的频率响应是严格限带的，且带限于折叠频率 $[-\Omega_s/2, \Omega_s/2]$ 以内时，才能使数字滤波器的频率响应在折叠频率以内，重现模拟滤波器的频率响应而不产生混叠失真。但是，任何一个实际的模拟滤波器频率响应都不是严格限带的，变换后都会产生周期延拓分量的频谱交叠，即产生频率响应的混叠失真，因而模拟滤波器的频率响应在折叠频率以上衰减越大、越快，变换后频率响应混叠失真就越小。

3．优缺点

冲激响应不变法有以下优缺点：

（1）冲激响应不变法使数字滤波器的冲激响应完全模仿模拟滤波器的冲激响应，即在时域逼近良好。

（2）模拟频率和数字频率之间呈线性关系，即 $\omega=\Omega T_s$，因而一个线性相位滤波器可以映射成一个线性相位的数字滤波器。

（3）由于有混叠效果，所以只适用于带限模拟滤波器，即只适用于低通滤波器和带通滤波器。对于高通和带阻滤波器不宜用冲激响应不变法。

【例 6-12】设计低通数字滤波器，要求在通带内频率低于 0.2πrad 时，允许幅度误差在 1dB 以内，在频率 0.3πrad～πrad 之间的阻带衰减大于 15dB。用脉冲响应不变法设计数字滤波器，T=1。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
wp=0.2*pi;                    %数字通带频率(弧度)
ws=0.3*pi;                    %数字阻带频率(弧度)
Rp=1;                         %通带波动(dB)
Rs=15;                        %阻带衰减(dB)
%模拟原型指标对频率的逆映射
```

```
T=1;    %置 T=1
OmegaP=wp/T;                    %原型通带频率
OmegaS=ws/T;                    %原型阻带频率
[cs,ds]=afd_cheb1(OmegaP,OmegaS,Rp,Rs)
[C,B,A]=sdir2cas(cs,ds);
[db,mag,pha,Omega]=freqs_m(cs,ds,pi);
subplot(234);plot(Omega/pi,mag);
title('模拟滤波器幅度响应|Ha(j\Omega)|');
[b,a]=imp_invr(cs,ds,T);
[h,n]=impz(b,a);
[C,B,A]=dir2par(b,a)
[db,mag,pha,grd,w]=freqz_m(b,a);
subplot(231);plot(w/pi,mag);
title('数字滤波器幅度响应|Ha(j \Omega)|');
subplot(232);plot(w/pi,db);
title('数字滤波器幅度响应(dB)');
subplot(233);plot(w/pi,pha/pi);
title('数字滤波器相位响应');
subplot(235);plot(n,h);
title('脉冲响应');
```

运行程序，输出如下，效果如图 6-12 所示。

```
***切比雪夫Ⅰ型滤波器的阶次= 4
cs =
    0.0383
ds =
    1.0000    0.5987    0.5740    0.1842    0.0430
C =
     []
B =
   -0.0833   -0.0246
    0.0833    0.0239
A =
    1.0000   -1.4934    0.8392
    1.0000   -1.5658    0.6549
```

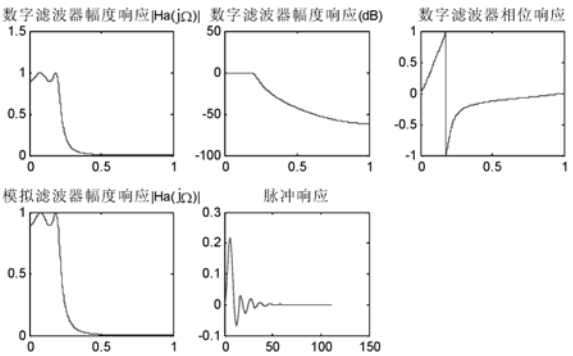


图 6-12　脉冲响应不变法设计数字滤波器

在运行程序过程中，调用到用户自定义编写的脉冲响应不变法子程序 imp_invr 函数。其源代码如下：

```
function [b,a]=imp_invr(c,d,T)
%脉冲响应不变法子程序
[R,p,k]=residue(c,d);
p=exp(p*T);
[b,a]=residuez(R,p,k);
b=real(b).*T;
a=real(a);
```

6.3.2　双线性变换法

1．变换原理

IIR 滤波器设计的另一简单、有效的方法就是双线性变换法。

该方法与前述冲激响应法的基本思路一样，不直接设计数字滤波器，而是先设计一个模拟 IIR 滤波器，然后映射成一个等效的数字滤波器。其变换原理如图 6-13 所示。

这样，就可以把 z 平面的数字滤波器的设计转化为 s 平面的等效模拟滤波器的设计。s 平面和 z 平面的映射关系为

$$s = f(z) = \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}} \tag{6-18}$$

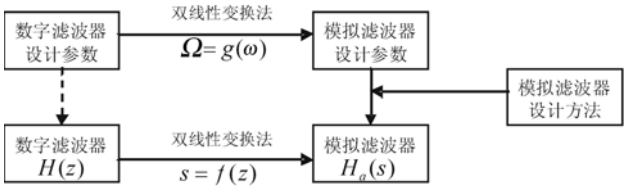


图 6-13　双线性变换法

将 $s = \mathrm{j}\Omega$ 及 $z = \mathrm{e}^{\mathrm{j}\omega}$ 代入式（6-18），得到数字频率与等效的模拟频率之间的映射关系为

$$\Omega = g(\omega) = \tan\left(\frac{\omega}{2}\right) \tag{6-19}$$

由于数字频率与模拟频率之间的变换关系不是线性关系，所以式（6-19）被称为频率预畸变换法。

双线性变换法的步骤如下：

（1）给定数字滤波器的幅度响应参数。

（2）用式（6-19）将数字滤波器参数变换为相应的等效模拟滤波器的参数。

（3）采用模拟滤波器设计方法设计等效模拟滤波器——H_a（s）。

（4）用式（6-18）把等效模拟滤波器逆映射为所期望的数字滤波器，即

$$H(z) = H_a(s)\big|_{s=f(z)} = H_a(f(z)) \tag{6-20}$$

$$H(\omega) = H_a(\Omega)\big|_{\Omega=g(\omega)} = H_a(g(\omega)) \tag{6-21}$$

2．优缺点

双线性变换法有以下优缺点：

（1）避免了频率响应的混叠现象。

（2）模拟频率与数字频率不再是线性关系，所以一个线性相位模拟滤波器经双线性变换后所得到的数字滤波器不再保持原有的线性相位了。

【例 6-13】设计低通数字滤波器，要求在通带内频率低于 0.2πrad 时，允许幅度误差在 1dB 以内，在频率 0.3πrad～πrad 之间的阻带衰减大于 15dB。用双线性变换法设计数字滤波器，T=1，模拟滤波器采用巴特沃思滤波器原型。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=0.2*pi;                    %数字通带频率(弧度)
ws=0.3*pi;                    %数字阻带频率(弧度)
Rp=1;                         %通带波动(dB)
Rs=15;                        %阻带衰减(dB)
%模拟原型指标对频率的逆映射
T=1;    %置 T=1
OmegaP=(2/T)*tan(wp/2);       %原型通带频率
OmegaS=(2/T)*tan(ws/2);       %原型阻带频率
[cs,ds]=afd_butt(OmegaP,OmegaS,Rp,Rs)
[C,B,A]=sdir2cas(cs,ds);
[db,mag,pha,Omega]=freqs_m(cs,ds,pi);
subplot(234);plot(Omega/pi,mag);
title('模拟滤波器幅度响应|Ha(j\Omega)|');
[b,a]=bilinear(cs,ds,T);          %双线性变换法设计
[h,n]=impz(b,a);
[C,B,A]=dir2par(b,a)
[db,mag,pha,grd,w]=freqz_m(b,a);
subplot(231);plot(w/pi,mag);
title('数字滤波器幅度响应|Ha(j\Omega)|');
subplot(232);plot(w/pi,db);
title('数字滤波器幅度响应(dB)');
subplot(233);plot(w/pi,pha/pi);
title('数字滤波器相位响应');
subplot(235);plot(n,h);
title('脉冲响应');
delta_w=2*pi/1000;
Rp=-(min(db(1:1:wp/delta_w+1)))
Rs=-round(max(db(Rs/delta_w+1:1:501)))
```

运行程序，输出如下，效果如图 6-14 所示。

```
  ***Butterworth Filter Order= 6
b =
      0.1480
a =
      1.0000    2.8100    3.9482    3.5168    2.0884    0.7862    0.1480
cs =
```

```
      0.1480
ds =
      1.0000    2.8100    3.9482    3.5168    2.0884    0.7862    0.1480
C =
      0.0092
B =
      1.8767   -0.4881
     -2.1781    1.0580
      0.2927   -0.4286
A =
      1.0000   -0.9459    0.2342
      1.0000   -1.0541    0.3753
      1.0000   -1.3143    0.7149
Rp =
      1.0000
Rs =
   18.0000
```

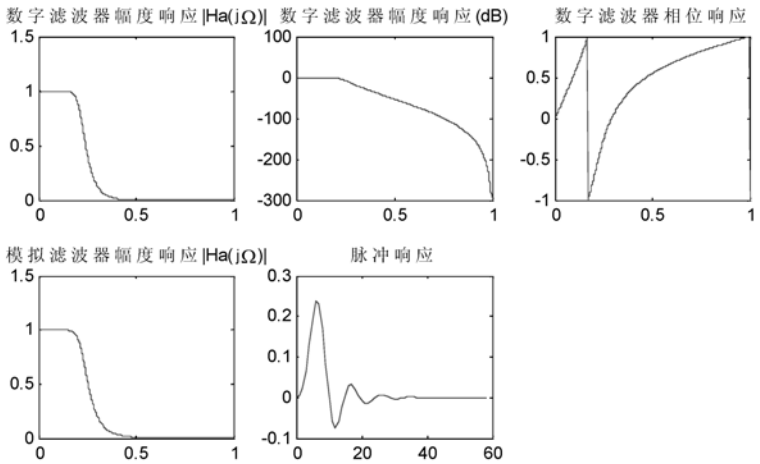


图 6-14　双线性变换法设计的巴特沃思型数字滤波器

在运行程序过程中，调用到用户自定义编写的数字滤波器响应子程序 freqz_m 函数。其源代码如下：

```
function [db,mag,pha,grd,w]=freqz_m(b,a)
% db 为相对振幅,mag 为绝对振幅,pha 为相位响应,grd 为群时延
% w 为频率样本点矢量,b 为 Ha(z)的分子多项式系数,a 为 Ha(z)的分母多项式系数
[h,w]=freqz(b,a,1000,'whole');
h=(h(1:501))';
w=(w(1:501))';
mag=abs(h);
db=20*log10((mag+eps)/max(mag));
pha=angle(h);
grd=grpdelay(b,a,w);
```

6.4 IIR 滤波器的设计方法

6.4.1 经典设计法

IIR 数字滤波器经典设计的一般步骤如下：

（1）根据给定的性能指标和方法不同，首先对设计性能指标中的频率指标，如数字边界频率进行变换，转换后的模拟频率指标作为模拟滤波器原型设计的性能指标。

（2）估计模拟滤波器最小阶数和边界频率，利用 MATLAB 工具箱提供的 buttord 函数、cheb1ord 函数、cheb2ord 函数、ellipord 函数等。

（3）设计模拟低通滤波器原型。利用 MATLAB 工具箱提供的 buttap 函数、cheb1ap 函数、cheb2ap 函数、ellipap 函数等。

（4）由模拟原型低通滤波器经频率变换获得模拟滤波器（低通、高通、带通、带阻等），利用 MATLAB 工具箱提供的 lp2lp 函数、lp2hp 函数、lp2bp 函数、lp2bs 函数。

（5）将模拟滤波器离散化获得 IIR 数字滤波器，利用 MATLAB 工具箱提供的 bilinear 函数或 impinvar 函数。

这里主要介绍第一步：关于设计性能指标的转换。

设计 IIR 滤波器时，给出的性能指标通常分数字指标和模拟指标两种。数字性能指标给出通带截止频率 ω_p，阻带起始频率 ω_s，通带波纹 R_p，阻带衰减 R_s 等。数字频率 ω_p 和 ω_s 的取值范围为 0～π，单位弧度。而 MATLAB 工具箱函数常采用归一化频率，ω_p 和 ω_s 的取值范围为 0～1，对应于 0～π，此时需进行转换。

模拟性能指标给出通带截止频率 Ω_p，阻带起始频率 Ω_s，通带波纹 R_p，阻带衰减 R_s 等。模拟频率 Ω_p 和 Ω_s 单位为 rad/s。

MATLAB 信号处理工具箱中，设计性能指标的转换应根据不同设计方法进行不同处理。下面就举例介绍这些方法。

【例 6-14】用脉冲响应不变法设计一个巴特沃思低通数字滤波器，使其特征逼近一个低通巴特沃思模拟滤波器的下列性能指标：通带截止频率Ω_p=2π×2000rad/s，通带波纹 R_p<3dB，阻带边界频率为 $\Omega_s = 2\pi \times 3000$rad/s，阻带衰减大于 15dB，采样频率 $F_s = 10000$Hz。假设一个信号 $x(t) = \sin 2\pi f_1 t + 0.5 * \cos 2\pi f_2 t$，其中 $f_1 = 1000$Hz，$f_2 = 4000$Hz。试将原信号与通过该滤波器的输出信号进行比较。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=2000*2*pi;   ws=3000*2*pi;          %滤波器截止频率
Rp=3;           Rs=15;                 %通带波纹和阻带衰减
Fs=10000;                              %采样频率
Nn=128;                                %调用 freqz 所用的频率点数
[N,wn]=buttord(wp,ws,Rp,Rs,'s');       %模拟滤波器的最小阶数
[z,p,k]=buttap(N);                     %设计模拟低通原型巴特沃思滤波器
[Bap,Aap]=zp2tf(z,p,k);                %将零点极点增益形式转换为传递函数形式
[b,a]=lp2lp(Bap,Aap,wn);               %进行频率转换
[bz,az]=impinvar(b,a,Fs);
```

```
% 运用脉冲响应不变法得到数字滤波器的传递函数
figure;
[h,f]=freqz(bz,az,Nn,Fs);              %绘制数字滤波器的幅频特性和相频特性
subplot(2,1,1); plot(f,20*log10(abs(h)));
xlabel('频率/Hz');   ylabel('振幅/dB');
grid on;
subplot(2,1,2);plot(f,180/pi*unwrap(angle(h)));
xlabel('频率/Hz');   ylabel('相位/^o');
grid on;
figure;
f1=1000;   f2=2000;                          %输入信号的频率
N=100;      %数据长度
dt=1/Fs;   n=0:N-1;                          %采样时间间隔
t=n*dt;    %时间序列
x=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t);         %滤波器输入信号
subplot(2,1,1);plot(t,x);
title('输入信号');                           %绘制输入信号
y=filtfilt(bz,az,x);                         %用函数 filtfilt 对输入信号进行滤波
y1=filter(bz,az,x);                          %用 filter 函数对输入信号进行滤波
subplot(2,1,2);plot(t,y,t,y1,':');
title('输出信号'); xlabel('时间/s');
legend('filtfilt 函数','filter 函数');       %加例说明
```

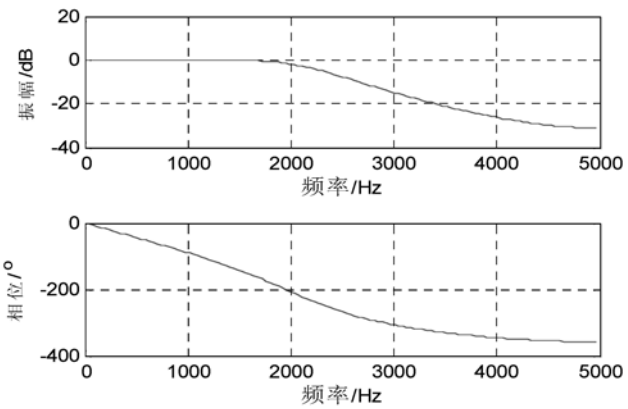


图 6-15　滤波器的频率响应

运行程序，效果如图 6-15 和图 6-16 所示。

由图 6-15 可知，在小于 2000Hz 处的衰减小于 3dB，而大于 3000Hz 处衰减大于 15dB，满足滤波器的设计指标。由图 6-16 可见滤波器对含有 1000Hz 和 4000Hz 频率成分的信号进行了滤波，滤除了 4000Hz 的信号。由程序的输出还可以看出，采用 filtfilt 函数，输出的 1000Hz 信号（实线）与输入 1000Hz 的信号相位一致，即经过滤波后并没有改变信号波形形状。而运用 filter 函数滤波后（虚线）有一些延迟，改变了信号的形状。

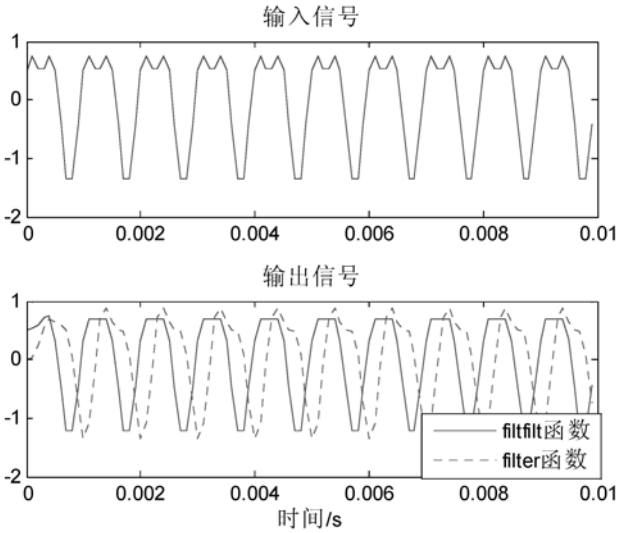


图 6-16　滤波器的输入和输出信号

【例 6-15】用双线性变换法设计一个椭圆低通滤波器，其性能指标为：通带截止频率 $0 \leqslant \omega \leqslant 0.2\pi$，通带波纹小于 1dB，阻带边界频率为 $0.3\pi \leqslant \omega \leqslant \pi$，幅度衰减大于 15dB，采样频率 T=0.01s。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=0.2*pi; ws=0.3*pi;                          %数字滤波器截止频率通带波纹
Rp=1; Rs=15;                                   %阻带衰减
Fs=100;    Ts=1/Fs;                            %采样频率
Nn=128;                                        %调用 freqz 所用的频率点数
wp=2/Ts*tan(wp/2);    ws=2/Ts*tan(ws/2);       %按频率公式进行转换
[n,wn]=ellipord(wp,ws,Rp,Rs,'s');              %计算模拟滤波器的最小阶数
[z,p,k]=ellipap(n,Rp,Rs);                      %设计模拟原型滤波器
[Bap,Aap]=zp2tf(z,p,k);                        %零点极点增益形式转换为传递函数形式
[b,a]=lp2lp(Bap,Aap,wn);                       %低通转换为低通滤波器的频率转换
[bz,az]=bilinear(b,a,Fs);                      %运用双线性变换法得到数字滤波器传递函数
[h,f]=freqz(bz,az,Nn,Fs);                      %绘出频率特性
subplot(2,1,1);plot(f,20*log10(abs(h)));
xlabel('频率/Hz');ylabel('振幅/dB');
grid on;
subplot(2,1,2);plot(f,180/pi*unwrap(angle(h)));
xlabel('频率/Hz');ylabel('相位/^o');
grid on;
```

运行程序，效果如图 6-17 所示。

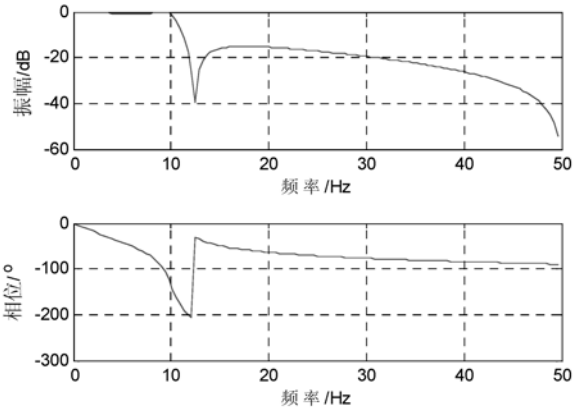


图 6-17　椭圆低通滤波器的频率特性

由图 6-17 可看出，在 10Hz 以前，衰减小于 1dB，在 15Hz 以后衰减均大于 15dB，即性能指标安全满足滤波器的设计要求。

6.4.2　直接设计法

IIR 数字滤波器的经典设计法只限于几种标准的低通、高通、带通、带阻滤波器，而对于任意形状或多频带的滤波器的设计是无能为力的。

如果所设计的 IIR 滤波器幅频特性比较复杂，可采用最小二乘法拟合给定幅频响应。使设计的滤波器幅频特性逼近期望的频率特性，这种方法称为 IIR 滤波器的直接设计法。

MATLAB 信号处理工具箱中提供 yulewalk 函数实现直接法设计 IIR 数字滤波器，其调用格式如下：

[b,a] = yulewalk(n,f,m)

式中：n 为滤波器的阶数；f 为给定的频率点向量，为归一化频率，取值范围为 0～1，f 的第一个频率点必须是 0，最后一个频率点必须为 1。其中 1 对应于奈奎斯特频率。在使用滤波器时，根据数据采样频率确定数字滤波器的通带和阻带在对此信号滤波的频率范围，根据数据采样频率确定数字滤波器的通带和阻带在对此信号滤波的频率范围。f 向量的频率点必须是递增的；m 为和频率向量 f 对应的理想幅值响应向量，m 和 f 必须是相同维数向量。b，a 分别是所设计滤波器的分子和分母多项式系数向量。IIR 滤波器的传递函数具有下面的形式：

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{a(1)+a(2)z^{-1}+\cdots+a(m+1)z^{-m}} \tag{6-22}$$

在定义频率响应时，应避免通带至阻带的过渡段形状过分尖锐。通常需要调用整过渡带的斜率来做到这点。

函数 yulewalk 首先计算给定幅频响应傅里叶逆变换和相关系数，再采用修正的 yule_walker 方程计算滤波器传递函数分母多项式系数。

函数 yulewalk 采取下面的步骤计算分子多项式：

（1）计算与分子多项式相应的幅值平方响应的辅助式。

（2）由辅助分子和分母多项式计算完全的频率响应。

（3）计算滤波器的脉冲响应。

（4）采用最小二乘法拟合脉冲响应最终求得滤波器的分子多项式系数。

注意：yulewalk 不能用来设计给定相位指标的滤波器。

【例 6-16】用直接法设计一个 10 阶多频带数字滤波器，幅频响应值如下：f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]，m=[0 0 1 1 0 0 1 1 1 0 0]，假设一个信号 x(t)=sin2*pi*f1*t+0.5*cos*2*pi*f2*t，其中 f1=6Hz，f2=17Hz。试将原信号与通过该滤波器的输出信号进行比较。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Order=10;                          %滤波器的阶数
f=0:0.1:1;                         %归一化频率点
m=[0 0 1 1 0 0 1 1 1 0 0];         %幅度点
[b,a]=yulewalk(Order,f,m);         %设计滤波器
[h,w]=freqz(b,a,128);              %计算 128 个点的频率特性
figure;
plot(f,m,'b-',w/pi,abs(h),'r:');   %绘制理想滤波器和设计滤波器的幅频特性
xlabel('归一化频率');ylabel('振幅');
title('运用 yulewalk 方法设计 IIR 滤波器');
legend('理想特性','实际设计',1);
figure
Fs=50;                             %信号采样频率成分
f1=6;   f2=17;                     %信号的频率成分
N=100;                             %数据点数
dt=1/Fs;   n=0:N-1;                %采样时间间隔
t=n*dt;                            %时间序列
x=sin(2*pi*f1*t)+0.5*cos(2*pi*f2*t);          %滤波器输入信号
subplot(2,1,1);plot(t,x);                     %绘制输入信号
title('输入信号');
y=filtfilt(b,a,x);                            %对信号进行滤波
subplot(2,1,2); plot(t,y);                    %绘制输出信号
title('输出信号'); xlabel('时间/s');
```

运行程序效果如图 6-18 和图 6-19 所示。

由图 6-18 可见，设计滤波器的幅频响应与理想滤波器的频率响应非常接近。当滤波器输入 6Hz 和 17Hz 的、以 50Hz 采样频率采样的信号后，输入信号的归一化频率为 6/(50/2)=0.24 和 17/(50/2)=0.68。由图 6-19 可见，0.24 和 0.68 均在通带范围内，因此这两个信号可以无阻碍地通过滤波器，因此输入信号和输出信号应该相同。图 6-19 验证了这一点。

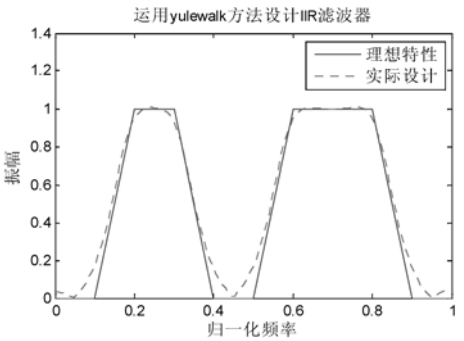


图 6-18　滤波器（虚线）及理想滤波器（实线）的幅频特性

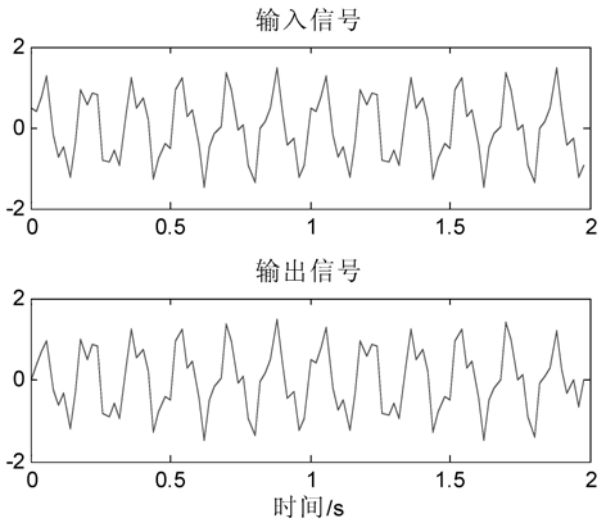


图 6-19　滤波器的输入和输出信号效果

6.5　高通滤波器的设计

6.5.1　模拟低通—数字高通变换

设计高通、带通、带阻等数字滤波器时，有两种方法：

方法一：先设计一个相应的高通、带通或带阻模拟滤波器，然后通过脉冲响应不变法或双线性变换法转换为数字滤波器，如图 6-20 所示。

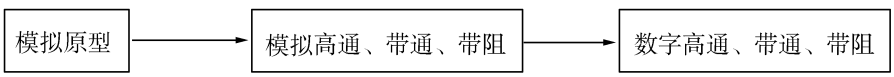


图 6-20　模拟原型转换为数字高带、带通、带阻

方法二：设计方法同 6.2 节讨论的低通滤波器的设计，如图 6-21 所示。

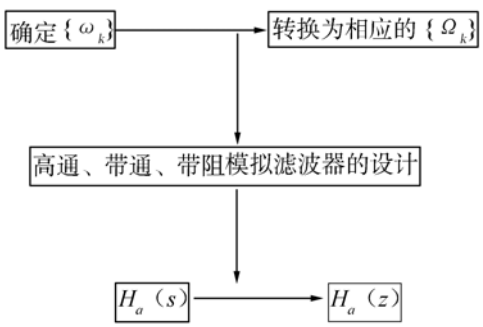


图 6-21　转换过程

第二种方法因其简捷便利，得到普遍采用。下面介绍变换方法的选用：

（1）脉冲响应不变法：对于高通、带阻等都不能直接采用，或只能在加了保护滤波器后才

可用。因此，使用直接频率变换（第二种方法），对脉冲响应不变法需要许多特殊的考虑，它一般应用于第一种方法中。

（2）双线性变换法：下面的讨论均用此方法，实际使用中多数情况也是如此。

下面介绍基于双线性变换法的高通滤波器的设计。

在模拟滤波器的高通设计中，低通至高通的变换就是 s 变量的倒置，这一关系同样可应用于双线性变换，只要将变换式的 s 代之以 1/s，就可得到数字高通滤波器，即

$$s=\frac{T1+Z^{-1}}{21-Z^{-1}} \tag{6-23}$$

由于倒数关系不改变模拟滤波器的稳定性，因此，也不会影响双线性变换后的稳定条件，而且 jΩ 轴仍映射在单位圆上，只是方向颠倒了。即

$$Z=\mathrm{e}^{\mathrm{j}\omega}\text{ 时，}\quad s=\frac{T1+\mathrm{e}^{\mathrm{j}\omega}}{21-\mathrm{e}^{\mathrm{j}\omega}}=\frac{T}{2}\mathrm{j}\cot\left(\frac{\omega}{2}\right)=\mathrm{j}\Omega \tag{6-24}$$

$$\Omega=-\frac{T}{2}\cot\left(\frac{\omega}{2}\right) \tag{6-25}$$

应当明确：所谓高通 DF，并不是 ω 高到 ∞，由于数字频域存在折叠频率 $\omega=\pi$ 对于实数响应的数字滤波器，ω 由π～2π部分只是 ω 由π～0 的镜像部分，因此有效的数字域仅是ω=π～0，高通也仅指这一段的高端，即到 $\omega=\pi$ 为止的部分。

高通变换的计算步骤和低通变换一样。

但在确定模拟原型预畸的临界频率时，应采用 $\Omega_k=\frac{T}{2}\cot\left(\frac{\omega_k}{2}\right)$ 不必加负号，因临界频率只有大小的意义而无正负的意义。

【例 6-17】设计一个巴特沃思高通滤波器，要求通带截止频率为 0.6π，通带内衰减不大于 1dB，阻带起始频率为 0.4π，阻带内衰减不小于 15dB，T=1。

```
wp=0.6*pi;ws=0.4*pi;
Rp=1; Rs=15;T=1;
[N,wn]=buttord(wp/pi,ws/pi,Rp,Rs)                %计算巴特沃思滤波器阶次和截止频率
[b,a]=butter(N,wn,'high');                       %频率变换法设计巴特沃思高通滤波器
[C,B,A]=dir2cas(b,a)
[db,mag,pha,grd,w]=freqz_m(b,a);
subplot(211);plot(w/pi,mag);
title('数字巴特沃思高通滤波器幅度响应|Ha(j\Omega)|');
subplot(212);plot(w/pi,db);
title('数字巴特沃思高通滤波器幅度响应(dB)');
```

运行程序结果如下，效果如图 6-22 所示。

```
N =     4
wn =    0.5344
C =     0.0751
B =
    1.0000   -2.0000    1.0000
    1.0000   -2.0000    1.0000
A =
    1.0000    0.1562    0.4488
    1.0000    0.1124    0.0425
```

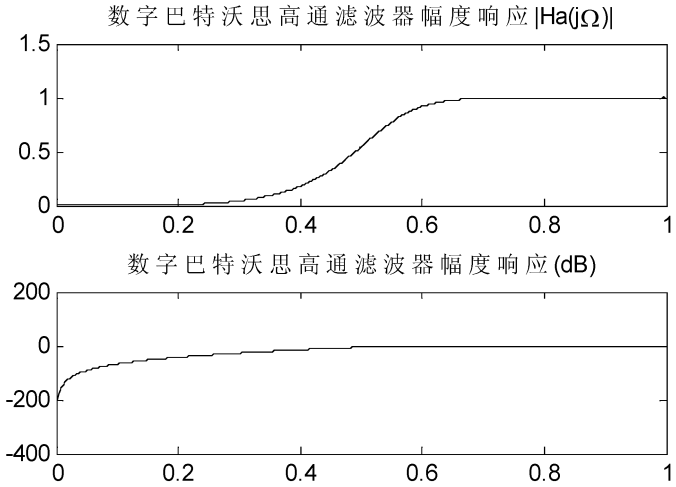


图 6-22　巴特沃思型数字高通滤波器

6.5.2　数字低通—数字高通变换

低通变换为高通只需将低通频率响应在单位圆上旋转 180°，即将 Z 变换成–Z，这就是旋转变换。只需将式 $H_d(z)=H_L(z)\big|_{z^{-1}=G(Z^{-1})}$ 中的 Z^{-1} 用 $-Z^{-1}$ 代替，就完成了数字低通到数字高通（Z）的变换，即

$$z^{-1}=\frac{-Z^{-1}-\alpha}{1+\alpha Z^{-1}} \tag{6-26}$$

变换关系式（6-26）的边界条件为

$$\begin{aligned} &z=1\rightarrow Z=-1\\ &z=\mathrm{e}^{\mathrm{j}\theta_c}\rightarrow Z=\mathrm{e}^{-\mathrm{j}\omega_c} \end{aligned} \tag{6-27}$$

将 $z=\mathrm{e}^{\mathrm{j}\theta_c}$，$Z=\mathrm{e}^{-\mathrm{j}\omega_c}$ 代入式（6-26）可得

$$\alpha=\frac{\cos\left(\dfrac{\theta_c+\omega_c}{2}\right)}{\cos\left(\dfrac{\theta_c-\omega_c}{2}\right)} \tag{6-28}$$

第 7 章　FIR 滤波器

在第 6 章中讨论了 IIR 滤波器的设计及其 MATLAB 实现，IIR 滤波器的优点是可利用模拟滤波器设计的结果，缺点是相位是非线性的，若需要线性相位，则要用全通网络进行校正。FIR 滤波器的优点是可方便地实现线性相位。

7.1　FIR 滤波器的结构

有限长冲激响应（FIR）滤波器有以下特点：

（1）系统的单位冲激响应 $h(n)$在有限个 n 值处不为零。

（2）系统函数 $H(z)$在$|z|>0$ 处收敛，极点全部在 $z=0$ 处（稳定系统）。

（3）结构上主要是非递归结构，没有输出到输入的反馈，但有些结构中（例如频率抽样结构）也包含有反馈的递归部分。

FIR 滤波器实现的基本结构如下：

（1）直接型结构。

（2）级联型结构。

（3）频率抽样型结构。

（4）快速卷积型结构。

7.1.1　直接型结构

设 FIR 滤波器的单位冲激响应 $h(n)$ 为一个长度为 N 的序列，则滤波器系统函数为

$$H(z)=\sum_{n=0}^{N-1}h(n)z^{-n} \tag{7-1}$$

表示这一系统输入/输出关系的差分方程为

$$y(n)=\sum_{n=0}^{N-1}h(m)x(n-m) \tag{7-2}$$

这就是 FIR 滤波器的直接型结构，又称卷积型结构。

7.1.2　级联型结构

FIR 级联型系统函数表示为

$$H(z)=b_0\prod_{k=1}^{K}(1+B_{k,1}z^{-1}+B_{k,2}z^{-2}) \tag{7-3}$$

即级联型 FIR 滤波器可以通过 casfilter 函数实现。但这种形式与 IIR 形式类似，也可以使用 dir2cas 函数，把分母向量 a 置为 1，用 cas2dir 函数从级联形式转换为直接形式而获得。

【例 7-1】FIR 滤波器的系统函数为

$$H(z)=\begin{cases}0.2^n, & 0\leqslant n\leqslant 5\\ 0, & \text{其他}\end{cases}$$

试分别用直接型和级联型实现。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=0:5;N=30;
b=0.2.^n;
delta=impseq(0,0,N);
h=filter(b,1,delta);
x=[ones(1,5),zeros(1,N-5)];
y=filter(b,1,x);
subplot(2,2,1);stem(h);
title('直接型 h(n)');
subplot(2,2,2);stem(y);
title('直接型 y(n)');
[b0,B,A]=dir2cas(b,1);
h=casfilter(b0,B,A,delta);
y=casfilter(b0,B,A,x);
subplot(2,2,3);stem(h);
title('级联型 h(n)');
subplot(2,2,4);stem(y);
title('级联型 y(n)');
```

运行程序，效果如图 7-1 所示。

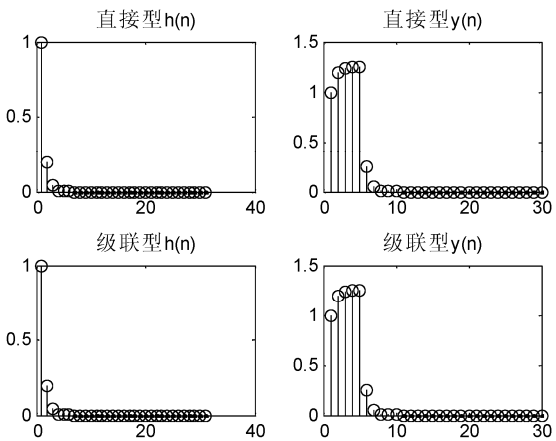


图 7-1　FIR 直接型与级联显示效果

7.1.3　频率抽样型结构

1．基本原理

若 FIR 滤波器的冲激响应为有限长（N 点）序列 $h(n)$，则有如图 7-2 所示的关系。

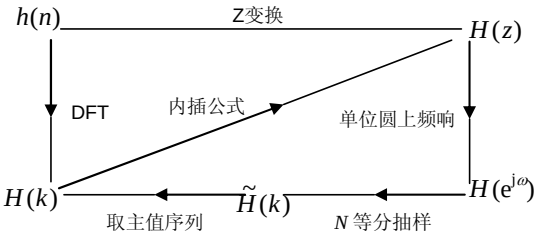


图 7-2　FIR 滤波器的冲激响应序列的关系图

因此，对 $h(n)$ 可以利用 DFT 得到 $H(k)$，然后利用内插公式

$$H(z)=(1-z^{-N})\frac{1}{N}\sum_{k=0}^{N-1}\frac{H(k)}{1-W_N^{-k}z^{-1}} \tag{7-4}$$

来表示系统函数。这就为 FIR 滤波器提供了另外一种结构——频率抽样结构。这种结构由两部分级联而成：

$$H(z)=\frac{1}{N}H_c(z)\sum_{k=0}^{N-1}H_k(z) \tag{7-5}$$

（1）级联的第一部分为

$$H_c(z)=1-z^{-N} \tag{7-6}$$

这是一个梳状滤波器，它滤掉了频率 $\omega=\frac{2\pi}{N}$ 及其各次谐波。

（2）级联的第二部分为 N 个一阶网络并联而成，第 k 个一阶网络为

$$H_k'(z)=\frac{H_k(z)}{1-W_N^{-k}z^{-1}} \tag{7-7}$$

它在单位圆上有一个极点：$z_k=W_N^{-k}=\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}k}$。

这是一个谐振频率 $\omega=\frac{2\pi}{N}$ 的无损耗谐振器。这个谐振器的极点正好与梳状滤波器的一个零点（i=k）相抵消，从而使这个频率上的频率响应等于 $H(k)$。这样，N 个谐振器的 N 个极点就和梳状滤波器的 N 个零点相抵消，从而在 N 个频率抽样点上的频率响应就分别等于 N 个 $H(k)$ 值。

2．修正

对 W_N^{-K} 和 $H(k)$ 是复数的情况，可以将第 k 个与第（N−k）个谐振器合并为一个实系数的二阶网络，将谐振器的实根、复根以及梳状滤波器合起来得到修正后的频率抽样型总结构。

（1）当 N 为偶数时的频率取样型表示如下：

$$H(z)=\frac{1-r^Nz^{-N}}{N}\left(\frac{H(0)}{1-rz^{-1}}+\frac{H(N/2)}{1-rz^{-1}}+\sum_{k=1}^{N/2-1}\frac{\beta_{0k}+\beta_{1k}z^{-1}}{1-z^{-1}2r\cos\left(\frac{2\pi}{N}k\right)+r^2z^{-2}}\right) \tag{7-8}$$

$$=\frac{1-r^Nz^{-N}}{N}\left(H_0(z)+H_{N/2}(z)+\sum_{k=1}^{N/2-1}H_k(z)\right)$$

其中：处于上下两端的是一阶节。

（2）当 N 为奇数时频率抽样型表示如下：

$$H(z)=\frac{1-r^N z^{-N}}{N}\left(\frac{H(0)}{1-rz^{-1}}+\frac{H(N/2)}{1-rz^{-1}}+\sum_{k=1}^{(N-1)/2-1}\frac{\beta_{0k}+\beta_{1k}z^{-1}}{1-z^{-1}2r\cos\left(\frac{2\pi}{N}k\right)+r^2z^{-2}}\right) \tag{7-9}$$

$$=\frac{1-r^N z^{-N}}{N}\left(H_0(z)+\sum_{k=1}^{(N-1)/2-1}H_k(z)\right)$$

其中：最上端的是一阶节。

【例 7-2】32 点线性相位 FIR 系统的频率样本定义如下：

$$|H(k)|=\begin{cases}0.5, & k=0,1,2\\ 1, & k=3\\ 0, & k=4,5,\cdots,15\end{cases}$$

求其频率抽样型结构，并比较它与线性相位形式计算的复杂度。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
M=32;
alpha=(M-1)/2;
magHk=[1 1 1 0.5 zeros(1,25) 0.5 1 1];
k1=0:15;
k2=16:M-1;
angHk=[-alpha*2*pi/M*k1,alpha*2*pi/M*(M-k2)];
H=magHk.*exp(j*angHk);
h=real(ifft(H,M));
[C,B,A]=dir2fs(h)
```

运行程序，输出如下：

```
C =
    2.0000
    2.0000
    1.0000
    0.0000
    0.0000
    0.0000
    0.0000
         0
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    1.0000
```

```
     0
B =
   -0.9952    0.9952
    0.9808   -0.9808
   -0.9569    0.9569
    0.0000   -0.7071
    0.1644    0.7288
   -0.4472   -0.6552
   -0.9487    0.4952
    1.0000   -0.0000
   -0.7071   -0.8315
    0.4472    0.9975
   -0.4472   -0.9921
    0.0000    0.7071
    0.0000   -0.5556
    0.9363    0.9994
    0.8000    0.6676
A =
   -1.9616    1.0000    1.0000
   -1.8478    1.0000    1.0000
   -1.6629    1.0000    1.0000
   -1.4142    1.0000    1.0000
   -1.1111    1.0000    1.0000
   -0.7654    1.0000    1.0000
   -0.3902    1.0000    1.0000
   -0.0000    1.0000    1.0000
    0.3902    1.0000    1.0000
    0.7654    1.0000    1.0000
    1.1111    1.0000    1.0000
    1.4142    1.0000    1.0000
    1.6629    1.0000    1.0000
    1.8478    1.0000    1.0000
    1.9616    1.0000    1.0000
    1.0000   -1.0000         0
    1.0000    1.0000         0
```

实现 32 点 FIR 系统频率抽样型结构时，注意到 H(0)=1，故一阶子系统不需要乘法运算，而 3 个二阶子系统中的每个子系统需要 3 次乘法运算，所以当每个样本经过有的二阶子系统时，共需要 9 次乘法运算，连同一阶子系统和二阶子系统总共需要 13 次加法运算。而实现相同点数的线性相位结构时，每个输出样本需要 16 次乘法和 31 次加法运算，因此 FIR 系统的频率抽样结构比线性相位结构复杂度更低，效率更高。

在运行程序过程中，调用用户自定义编写实现 FIR 系统直接型结构转换为频率取样型结构的函数 dir2fs，其源代码如下：

```
function [C,B,A]=dir2fs(h)
% 直接型到频率采样型的转换
% C 为包含各并行部分增益的行向量,B 为包含按行排列的分子系数矩阵
% A 为包含按行排列的分母系数矩阵,h 为 FIR 滤波器的脉冲响应向量
```

```
M=length(h);
H=fft(h,M);
magH=abs(H); phaH=angle(H)';
if (M==2*floor(M/2))
    L=M/2-1;              %M 为偶数
    A1=[1 -1 0;1 1 0];
    C1=[real(H(1)),real(H(L+2))];
else
    L=(M-1)/2;
    C1=[real(H(1))];
end
k=[1:L]';
% 初始化 B 和 A 数组
B=zeros(L,2);   A=ones(L,3);
% 计算分母系数
A(1:L,1)=-2*cos(2*pi*k/M); A=[A;A1];
% 计算分子系数
B(1:L,1)=cos(phaH(2:L+1));
B(1:L,2)=-cos(phaH(2:L+1)-(2*pi*k/M));
% 计算增益系数
C=[2*magH(2:L+1),C1]';
```

7.1.4　快速卷积型结构

若 FIR 滤波器的单位冲激响应 $h(n)$ 是一个 N_1 点有限长序列，输入 $x(n)$ 是一个 N_2 点有限长序列，那么输出 $y(n)$ 是 $h(n)$ 与 $x(n)$ 的线性卷积，它是一个 $L=N_1+N_2=1$ 点的有限长序列。

我们知道，将 $x(n)$ 补上 $L-N_2$ 个零值点，将 $h(n)$ 补上 $L-N_1$ 个零值点，然后进行 L 点圆周卷积，就可以代替原 $x(n)$ 与 $h(n)$ 的线性卷积。

而圆周卷积可以用 DFT 和 IDFT 的方法来计算，这样得到 FIR 滤波器的快速卷积结构，如图 7-3 所示。

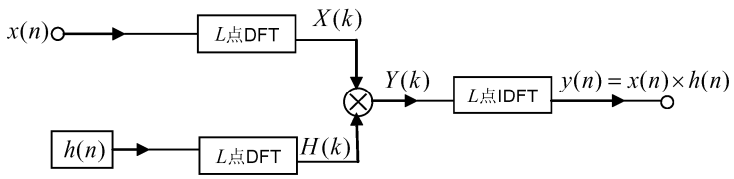


图 7-3　FIR 滤波器快速卷积结构

7.2　线性相位 FIR 数字滤波器的特性

在很多实际应用中，例如语音或音频信号处理中，数字滤波器常常被用来实现选频操作，因此，通常把频域幅度和相位响应作为性能指标，而在通带中，常常希望系统具有线性相位响应。

幅度指标可以按两种方式给出。第 1 种是绝对指标，它提供对幅度响应函数 $|H(e^{j\omega})|$ 的要求，这些指标一般应用于 FIR 滤波器的设计。IIR 滤波器设计则以一种不同的方式给出指标。

第 2 种指标是相对指标，它以分贝（dB）值的形式提出要求，其值定义为如下形式：

$$\mathrm{dB}=-20\lg\frac{|H(\mathrm{e}^{\mathrm{j}\omega})|}{|H(\mathrm{e}^{\mathrm{j}\omega})|_{\max}} \tag{7-10}$$

7.2.1 线性相位 FIR 滤波器幅度特性

本小节介绍线性相位滤波器的冲激响应和频率响应的形状及其系统函数的零极点的位置。设 $h(n)$，$0\leqslant n\leqslant M-1$ 是长度（或持续时间）为 M 的冲激响应，那么系统函数可表示为

$$H(z)=\sum_{n=0}^{M-1}h(n)z^{-n}=z^{-(M-1)}\sum_{n=0}^{M-1}h(n)z^{M-1-n} \tag{7-11}$$

式（7-10）在原点 $z=0$ 处有（$M-1$）重极点和（$M-1$）个位于 z 平面上任意位置的零点。其频率响应为

$$H(\mathrm{e}^{\mathrm{j}\omega})=\sum_{n=0}^{M-1}h(n)\mathrm{e}^{-\mathrm{j}\omega n},\quad -\pi<\omega\leqslant\pi \tag{7-12}$$

1．冲激响应

给系统加一个线性相位约束，满足

$$\angle H(\mathrm{e}^{\mathrm{j}\omega})=\beta-\alpha\omega,\quad -\pi<\omega\leqslant\pi \tag{7-13}$$

冲激响应可根据其对称性分为两类：对称冲激响应和反对冲激响应。

1）对称冲激响应

满足式（7-13）的称为冲激响应。可分为两种：

$$h(n)=h(M-1-n),\quad \beta=0,0\leqslant n\leqslant M-1 \tag{7-14}$$

（1）M 为奇数，在此情况下 $\alpha=(M-1)/2$ 为整数。

（2）M 为偶数，在此情况下 $\alpha=(M-1)/2$ 不为整数。

2）反对称冲激响应

$$h(n)=-h(M-1-n),\quad \beta=\pm\pi/2,0\leqslant n\leqslant M-1 \tag{7-15}$$

（1）M 为奇数，在此情况下 $\alpha=(M-1)/2$ 为整数。

（2）M 为偶数，在此情况下 $\alpha=(M-1)/2$ 不为整数。

2．频率响应

把 M 为奇数和偶数的情形与对称和反对称组合在一起，即可以得到 4 种类型的线性相位 FIR 滤波器，见表 7-1。每种情况下的频率响应函数具有特定的表示和形状。为了研究这些响应，把 $H(\mathrm{e}^{\mathrm{j}\omega})$ 改写为

$$H(\mathrm{e}^{\mathrm{j}\omega})=H_r\mathrm{e}^{\mathrm{j}(\beta-\alpha\omega)},\quad \beta=\pm\frac{\pi}{2},\alpha=\frac{M-1}{2} \tag{7-16}$$

式中：$H(\mathrm{e}^{\mathrm{j}\omega})=H_r(\omega)$ 为振幅响应而不是幅度响应函数。振幅响应函数是实函数，它既可以是正的也可以是负的，而幅度响应则永远是正的，与幅度响应有关的相位响应是一个不连续的函数，而与振幅响应有关的则是一个连续的函数。

表 7-1　线性相位 FIR 滤波器类型

类　　型	说　　明	条　　件
1 型线性相位 FIR 滤波器	对称冲激响应，M 为奇数	$\beta=0, a=(M-1)/2$ 为整数，$h(n)=h(M-1-n)$
2 型线性相位 FIR 滤波器	对称冲激响应，M 为偶数	$\beta=0, a=(M-1)/2$ 不为整数，$h(n)=h(M-1-n)$
3 型线性相位 FIR 滤波器	反对称冲激响应，M 为奇数	$\beta=0, a=(M-1)/2$ 为整数，$h(n)=-h(M-1-n)$
4 型线性相位 FIR 滤波器	反对称冲激响应，M 为偶数	$\beta=0, a=(M-1)/2$ 不为整数，$h(n)=-h(M-1-n)$

为了实现线性相位滤波器振幅响应，用户自定义编写 4 种类型函数来计算线性相位滤波器振幅响应。其源代码如下：

```
%1 型线性相位滤波器振幅响应
function [hr,w,a,L]=hr_type1(h)
%  计算 1 型低通滤波器设计的振幅响应 hr(w)
% hr 为振幅响应
% w 为[0,pi]区间计算 hr 的 500 个频率点
% a 为 1 型低通滤波器的系数
% L 为 hr 的阶次
% h 为 1 型低通滤波器的冲激响应
M=length(h);
L=(M-1)/2;
a=[h(L+1) 2*h(L:-1:1)];
n=[0:1:L];
w=[0:1:500]'*pi/500;
hr=cos(w*n) *a';
```

```
%2 型线性相位滤波器振幅响应
function [hr,w,b,L]= hr_type2 (h)
%  计算 2 型低通滤波器设计的振幅响应 hr(w)
% hr 为振幅响应
% w 为[0,pi]区间计算 hr 的 500 个频率点
% b 为 2 型低通滤波器的系数
% L 为 hr 的阶次
% h 为 2 型低通滤波器的冲激响应
M=length(h);
L=M/2;
b=2*h(L:-1:1);
n=[1:1:L];
n=n-0.5;
w=[0:1:500]'*pi/500;
hr=cos(w*n)*b';
```

```
%3 型线性相位滤波器振幅响应
function [hr,w,c,L]= hr_type3(h)
%  计算 3 型低通滤波器设计的振幅响应 hr(w)
% hr 为振幅响应
% w 为[0,pi]区间计算 hr 的 500 个频率点
```

```
% c 为 3 型低通滤波器的系数
% L 为 hr 的阶次
% h 为 3 型低通滤波器的冲激响应
M=length(h);
L=(M-1)/2;
c=[2*h(L+1:-1:1)];
n=[0:1:L];
w=[0:1:500]'*pi/500;
hr=sin(w*n)*c';
```

```
%4 型线性相位滤波器振幅响应
function [hr,w,d,L]= hr_type4(h)
%  计算 4 型低通滤波器设计的振幅响应 hr(w)
% hr 为振幅响应
% w 为[0,pi]区间计算 hr 的 500 个频率点
% d 为 4 型低通滤波器的系数
% L 为 hr 的阶次
% h 为 4 型低通滤波器的冲激响应
M=length(h);
L=M/2;
d=2*[h(L:-1:1)];
n=[1:1:L];
n=n-0.5;
w=[0:1:500]'*pi/500;
hr=sin(w*n)*d';
```

3．零点设计

对于线性相位 FIR 滤波器而言，由于 $h(n)$ 具有对称性约束，其零点也具有一定的对称性。由 DSP 理论可知，如果 $H(z)$ 在

$$z = z_1 = re^{j\theta}$$

处有一个零点，则根据线性相位特性，在

$$z = \frac{1}{z_1} = \frac{1}{r}e^{-j\theta}$$

处必定有一个零点。对于一个实数滤波器，如果 z_1 为复数，则在 $z_1^* = re^{-j\theta}$ 处有一个共轭零点，这就意味着在 $1/z_1^* = (1/r)e^{j\theta}$ 处也有一个零点，因此，一般零点的构造是一个四套件：

$$re^{j\theta}；\ \frac{1}{r}e^{j\theta}；\ re^{-j\theta}；\ \frac{1}{r}e^{-j\theta}$$

在此用户先自定义 pzkplo 函数，其功能在于绘制数字滤波器的零极点图，这个函数是用户自己编写的，下面例子将应用到，其源程序如下：

```
function pzkplot(num,den)
%绘制系统函数的零极点图
% pzkplot(num,den)
% num 为系统函数分子多项式系数向量
% den 为系统函数分母多项式系数向量
hold on
```

```
axis('square');
%绘制单位元
x=-1:0.01:1;
y=(1-x.^2).^0.5;
y1=-(1-x.^2).^0.5;
plot(x,y,'b',x,y1,'b');
%求取系统的零极点
num1=length(num);
den1=length(den);
if (num1>1),
    z=roots(num);
else
    z=0;
end
if (den1>1),
    p=roots(den);
else
    p=0;
end
%判断绘图范围
if (num>1 & den1>1)
    r_max_z=max(abs(real(z)));
    i_max_z=max(abs(imag(z)));
    a_max_z=max(r_max_z,i_max_z);
    r_max_p=max(abs(real(p)));
    i_max_p=max(abs(imag(p)));
    a_max_p=max(r_max_p,i_max_p);
    a_max=max(a_max_z, a_max_p);
elseif (num1>1)
    r_max_z=max(abs(real(z)));
    i_max_z=max(abs(imag(z)));
    a_max=max(r_max_z,i_max_z);
else
     r_max_p=max(abs(real(p)));
     i_max_p=max(abs(imag(p)));
     a_max=max(r_max_p, i_max_p);
end
%确定绘图范围，并绘制出轴线和边框线
axis([-a_max a_max -a_max a_max]);
plot([-a_max a_max],[0 0],'b');
plot([0 0],[-a_max a_max],'b');
plot([-a_max a_max],[a_max a_max],'b');
plot([a_max a_max],[-a_max a_max],'b');
%绘制出零极点
Lz=length(z);
for i=1:Lz
    plot(real(z(i)),imag(z(i)),'bo');
end
```

```
Lp=length(p);
for j=1:Lp
    plot(real(p(j)),imag(p(j)),'bx');
end
title('The zeros-pole plot');
xlabel('虚部');
ylabel('实部');
```

【例 7-3】已知滤波器的系统函数为

$$h(n)=\{\overline{-5},2,-3,-1,7,-1,-3,2,-5\}$$

要求求出振幅响应 $H_r(\omega)$ 以及零点位置。

在 M 文件编辑器中输入以下代码：

```
clear;
h=[-5 2 -3 -1 7 -1 -3 2 -5];
M=length(h);
n=0:M-1;
[hr,w,a,L]=type1(h);
a,L
a_max=max(a)+1;
a_min=min(a)-1;
subplot(221),
stem(n,h);
axis([-1 2*L+1 a_min a_max]);
ylabel('h(n)');
title('冲激响应');
subplot(222),
stem(0:L,a);
axis([-1 2*L+1 a_min a_max]);
ylabel('a(n)');
title('a(n)系数');
subplot(223),
plot(w/pi,hr);
grid on;
xlabel('以 pi 为单位的频率'); ylabel('hr');
title('1 型振幅响应');
subplot(224),
pzkplot(h,1);
```

运行程序，输出结果如下：

```
a =
     7     -2     -6      4    -10
L =
     4
```

同时得到如图 7-4 所示的效果图。

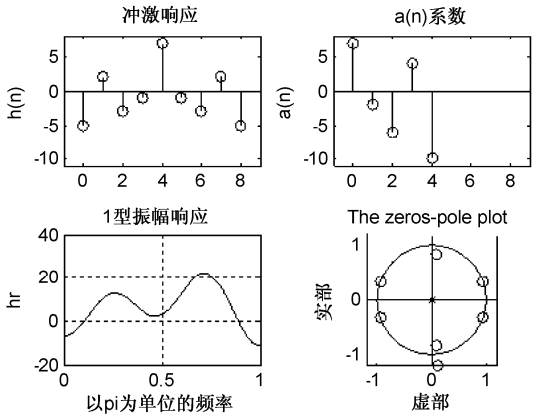


图 7-4　1 型线性相位滤波器的性能

7.2.2　线性相位 FIR 滤波器零点特性

由于线性相位 FIR 滤波器的单位冲激响应具有对称性，即

$$h(n)=\pm h(N-1-n),\quad +,-\text{对应奇偶对称} \tag{7-17}$$

经 $m=M-1-n$ 置换可得

$$H(z)=\pm z^{-(N-1)}\sum_{n=0}^{M-1}h(n)(z^{-1})^{-n}=\pm z^{-(N-1)}H(z^{-1}) \tag{7-18}$$

由式（7-18）可看出，若 $z=z_i$ 是 $H(z)$ 的零点，则 $z=z_i^{-1}$ 也一定是 $H(z)$ 的零点。由于 $h(n)$ 是实数，$H(z)$ 的零点还必须共轭成对，所以 $z=z_i^*$ 及 $z=1/z^*$ 也必是零点。

因此，线性相位滤波器的零点必须是互为倒数的共轭对，即成四对出现，这种共轭对共有四种可能的情况（四个不同的零点结构）：

（1）既不在单位圆上，也不在实轴上，有四个互为倒数的两组共轭对：z_i，z_i^*；$1/z_i$，$1/z_i^*$，如图 7-5 所示。

（2）在单位圆上，但不在实轴上，因倒数等于其共轭，有一对共轭零点：z_i，z_i^*，如图 7-6 所示。

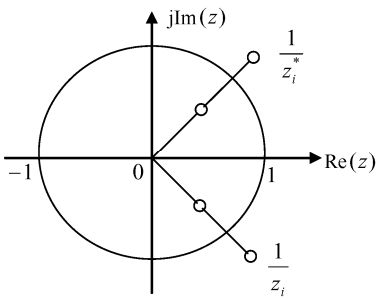


图 7-5　有四个互为倒数的两组共轭对

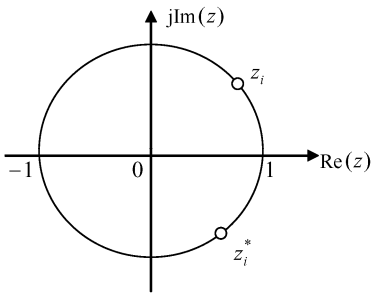


图 7-6　因倒数等于其共轭，有一对共轭零点

（3）不在单位圆上，但在实轴上，共轭是其本身，有一对互为倒数的零点：z_i，$1/z_i$，如图 7-7 所示。

（4）既在单位圆上，又在实轴上，共轭和倒数都合为一点，所以成单出现，只有两种可能，

z_i=1 或 $z_i=-1$，如图 7-8 所示。

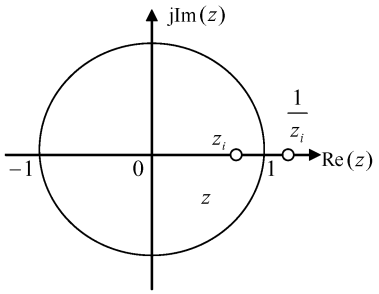


图 7-7　共轭是其本身，有一对互为倒数的零点

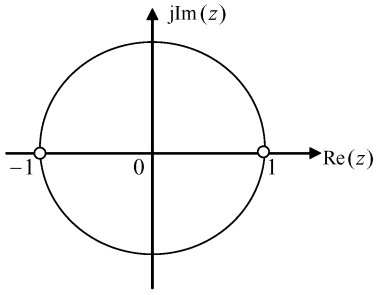


图 7-8　共轭和倒数都合为一点，成单出现

从幅度响应的讨论中已经知道，对于第二种 FIR 滤波器（$h(n)$ 偶对称，N 为偶数），$H(\pi)=0$，即 $z=e^{j\pi}=-1$，$z=e^{j\pi}=-1$ 是 $H(\omega)$ 的零点，既在单位圆，又在实轴，所以必有单根；同样道理，对于第三种 FIR 滤波器，$h(n)$ 奇对称，N 为奇数，因 $H(0)=0$，$H(\pi)=0$，所以 $z=1$ 和 $z=-1$ 都是 $H(z)$ 的单根。

所以，$h(n)$ 奇对称 $\rightarrow H(0)=0$，N 为偶数 $\rightarrow H(\pi)=0$。

线性相位滤波器是 FIR 滤波器中最重要的一种，应用最广。实际使用时应根据需要选择其合适类型，并在设计时遵循其约束条件。

7.3　基本窗函数法的 FIR 滤波器设计

7.3.1　窗函数的原理

FIR 滤波器设计的主要任务是根据给定的性能指标确定滤波器的系数 b，即系统单位脉冲序列 $h(n)$，它是一个有限长序列。

FIR 滤波器的理想频率响应，可写成复数形式的傅里叶级数形式：

$$H_d(e^{j\omega})=\sum_{n=-\infty}^{+\infty}h_d(n)e^{-j\omega n} \tag{7-19}$$

式中：$h_d(n)$ 是对应的单位脉冲响应序列。这说明滤波器的频率响应和单位脉冲响应互为傅里叶变换对。因此其单位脉冲响应可由下式求得：

$$h_d(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(e^{j\omega n})d\omega \tag{7-20}$$

求得序列 $h_d(n)$ 后，通过 Z 变换，可得到 $H_d(z)$，即

$$H_d(z)=\sum_{n=-\infty}^{+\infty}h_d(n)z^{-n} \tag{7-21}$$

注意，这里 $h_d(n)$ 为无限长序列，因此 $H_d(z)$ 是物理上不可实现的。如何变成物理上可实现呢？一个自然的想法是只取其中的某些项，即只截取 $h_d(n)$ 中的一部分，比如 $n=0,\cdots,N-1$，N 为正整数。这种处理相当于将 $h_d(n)$，$n=-\infty\sim+\infty$ 与函数 $\omega(n)$ 相乘，$\omega(n)$ 具有下列形式：

$$\omega(n)=\begin{cases}0, & n<0, n\geqslant N\\ 1, & 0\leqslant n<N\end{cases}$$

$\omega(n)$ 相当于一个矩形，称之为矩形窗。即可采用矩形窗函数 $\omega(n)$ 将无限脉冲响应 $h_d(n)$ 截取一段 $h(n)$ 来近似为 $h_d(n)$，这种截取在数学上表示为

$$h(n)=h_d(n)\omega(n) \tag{7-22}$$

这里应该强调的是，加窗函数不是可有可无的，而是将设计变为物理可实现所必需的。

截取之后的滤波器传递函数变为

$$H(z)=\sum_{n=0}^{N-1}h(n)z^{-n} \tag{7-23}$$

式中：N 为窗口宽度；$H(z)$ 是物理可实现系统。

为了获得线性相位，FIR 滤波器 $h(n)$ 必须满足中心对称条件，序列 $h(n)$ 的延迟为 $\alpha=(N-1)/2$。

这种方法的基本原理是用一定宽度的矩形窗函数截取无限脉冲响应序列，获得有限长的脉冲响应序列，从而得到 FIR 滤波器的脉冲响应，故称为 FIR 滤波器的窗函数设计法。

理想频响也可以写成幅度函数和相位函数的形式：

$$H_d(\mathrm{e}^{\mathrm{j}\omega})=H_d(\omega)\mathrm{e}^{-\mathrm{j}\omega\alpha} \tag{7-24}$$

其中幅度函数为

$$H_d(\omega)=\begin{cases}1, & |\omega|\leqslant\omega_c\\ 0, & \omega_c\leqslant|\omega|\leqslant\pi\end{cases} \tag{7-25}$$

两个信号时域乘积对应于频域卷积，所以

$$H(\mathrm{e}^{\mathrm{j}\omega})=H_d(\mathrm{e}^{\mathrm{j}\omega})*W_R(\mathrm{e}^{\mathrm{j}\omega})=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\mathrm{e}^{\mathrm{j}\omega n})W_R[\mathrm{e}^{\mathrm{j}(\omega-\theta)}]\mathrm{d}\theta$$

$$=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\mathrm{e}^{\mathrm{j}\omega})\mathrm{e}^{-\mathrm{j}\omega\alpha}W_R(\omega-\theta)\mathrm{e}^{-\mathrm{j}(\omega-\theta)\alpha}\mathrm{d}\theta$$

$$=\mathrm{e}^{-\mathrm{j}\omega\alpha}\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\mathrm{e}^{\mathrm{j}\omega n})W_R(\omega-\theta)\mathrm{d}\theta\right]$$

如果也以幅度函数和相位函数来表示 $H(\mathrm{e}^{\mathrm{j}\omega})$，即

$$H(\mathrm{e}^{\mathrm{j}\omega})=H(\omega)\mathrm{e}^{-\mathrm{j}\omega\alpha}$$

则实际 FIR 滤波器的幅度函数 $H(\omega)$ 为

$$H(\mathrm{e}^{\mathrm{j}\omega})=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\theta)W_R(\omega-\theta)\mathrm{d}\theta \tag{7-26}$$

正好是理想滤波器幅度函数与窗函数幅度函数的卷积。

加窗对理想频响的影响：

（1）使理想频响不连续边沿加窗，形成过渡带，过渡带的宽度等于 $W_R(\omega)$ 的主瓣宽度，与 N 成反比。

（2）过渡带两旁产生肩峰和余振。肩峰和余振的大小取决于 $W_R(\omega)$ 的副瓣，副瓣多，余振多；副瓣相对值大，余振强，与 N 无关。

$$W_R(\omega)=\frac{\sin(\omega N/2)}{\sin(\omega/2)}=N\frac{\sin(\omega N/2)}{\omega/2}\approx N\frac{\sin(x)}{x} \tag{7-27}$$

其中：$x=\omega N/2$，所以 N 的改变不影响主瓣与旁瓣的比例关系，最多只能改变 $W_R(\omega)$ 的绝对值大小和起伏的密度。当 N 增大时，$W_R(\omega)$ 的幅值变大，频率轴变密，而最大肩峰经计算可知始终 8.95%，这种现象称为吉布斯（Gibbs）效应。

肩峰值的大小决定了滤波器通带的平稳程度和阻带的衰减，对滤波器的性能有很大的影响。

为了改善滤波器的特性，必须改变窗函数的形状，窗函数要满足以下两点要求：

（1）窗谱主瓣宽度要窄，以获得较陡的过渡带；

（2）相对于主瓣幅度，旁瓣要尽可能小，使能量尽量集中在产瓣中，这样就可以减小肩峰和余振，以提高阻带衰减和通带平稳性。

但实际上这两点不能兼得，一般总是通过增加主瓣宽度来换取对旁瓣的抑制。

7.3.2 矩形窗

前面讲解原理时应用的就是矩形窗。

在 MATLAB 中，实现矩形窗的函数为 boxcar 和 rectwin，函数调用格式如下：

w = rectwin(L)

其中：N 时窗函数的长度，返回值 w 是一个 N 阶的向量，它的元素由窗函数的值组成。其中 w=boxcar 等价于 w=ones(N, 1)。

其中 boxcar 函数被 fwind1 函数代替。此函数的调用格式如下：

h = fwind1(Hd, win)

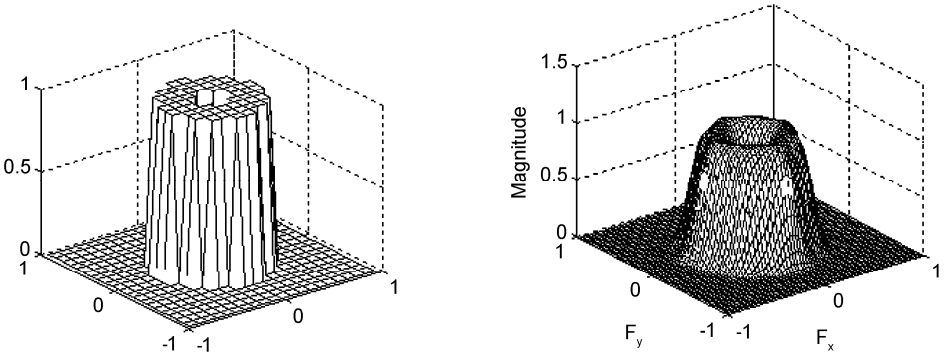
h = fwind1(Hd, win1, win2)

h = fwind1(f1, f2, Hd,...)

【例 7-4】矩形窗的实现。

```
>> [f1,f2] = freqspace(21,'meshgrid');
Hd = ones(21);
r = sqrt(f1.^2 + f2.^2);
Hd((r<0.1)|(r>0.5)) = 0;
colormap(jet(64))
mesh(f1,f2,Hd)
 h = fwind1(Hd,hamming(21));
freqz2(h)
```

运行程序，输出效果如图 7-9 所示。



（a）三维曲线图　　（b）频率响应图

图 7-9　矩形窗的显示效果

7.3.3　汉宁窗

在 MATLAB 信号处理工具箱中，提供了 hanning 或 barthannwin 函数实现汉宁窗，其调用格式如下：

w=hanning(L)

w = barthannwin(L)

汉宁窗的表达式如下：

$$w(n)=\frac{1}{2}\left[1-\cos\left(\frac{2\pi n}{N-1}\right)\right]R_W(n)=0.5R_N(n)-0.25\left(\mathrm{e}^{\mathrm{j}\frac{2\pi n}{N-1}}+\mathrm{e}^{-\mathrm{j}\frac{2\pi n}{N-1}}\right)R_N(n) \tag{7-28}$$

利用傅里叶变换的移位特性，汉宁窗频谱的幅度函数 $W(\omega)$ 可用矩形窗的幅度函数表示为

$$W(\omega)=0.5W_R(\omega)-0.25\left[W_R\left(\omega-\frac{2\pi n}{N-1}\right)+W_R\left(\omega+\frac{2\pi n}{N-1}\right)\right] \tag{7-29}$$

三部分矩形窗频谱相加，使旁瓣互相抵消，能量集中在主瓣，旁瓣大小减小，主瓣宽度增加 1 倍。

【例 7-5】绘制汉宁窗的频响。

```
>> clear all;
L=64;
wvtool(barthannwin(L))
```

运行程序，效果如图 7-10 所示。

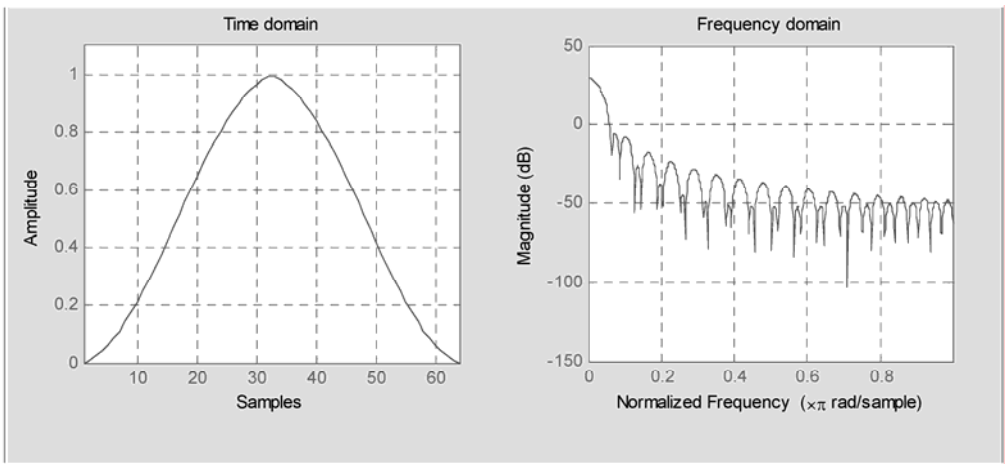


图 7-10　汉宁窗频响

【例 7-6】利用希伯尔特变换器设计汉宁窗。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
M=25; alpha=(M-1)/2;
n=0:M-1;
hd=(2/pi)*((sin((pi/2)*(n-alpha)).^2)./(n-alpha));
hd(alpha+1)=0;
```

```
w_han=(hanning(M))';
h=hd.*w_han;
[Hr,w,P,L]=hr_type3(h);
subplot(2,2,1);stem(n,hd);
title('理想脉冲响应');
axis([-1 M -1.2 1.2]);
ylabel('hd(n)'); text(M+1,-1.2,'n');
subplot(2,2,2);stem(n,w_han);
title('汉宁窗');
axis([-1 M 0 1.2]);
ylabel('w(n)'); text(M+1,-1.2,'n');
subplot(2,2,3);stem(n,h);
title('实际脉冲响应');
axis([-1 M -1.2 1.2]);
xlabel('n');ylabel('h(n)');
w=w'; Hr=Hr';
w=[-fliplr(w),w(2:501)];
Hr=[-fliplr(Hr),Hr(2:501)];
subplot(2,2,4);plot(w/pi,Hr);
title('振幅响应');
grid on;
xlabel('频率/pi');ylabel('Hr');
axis([-1 1 -1.1 1.1]);
set(gca,'XTickMode','manual','XTick',[-1 0 1]);
set(gca,'YTickMode','manual','YTick',[-1 0 1]);
```

运行程序，效果如图 7-11 所示。

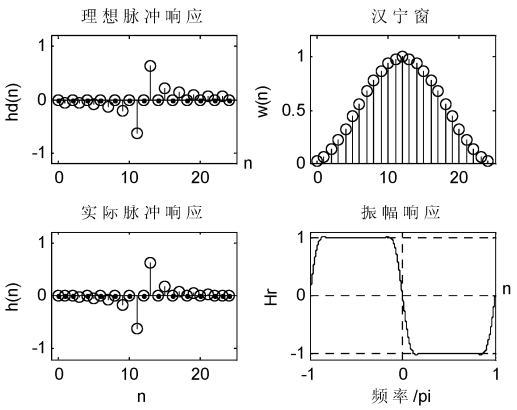


图 7-11　希伯尔特变换器设计的汉宁窗效果

7.3.4　海明窗

在 MATLAB 信号处理工具箱中，提供了 hamming 函数实现海明窗，其调用格式如下：

w = hamming(L)

w = hamming(L,'sflag')

海明窗的表达式为

$$w(n)=\left[0.54-0.46\cos\left(\frac{2\pi n}{N-1}\right)\right]R_N(n) \qquad (7\text{-}30)$$

它是对汉宁窗的改进，在主瓣宽度（对应第一零点的宽度）相同的情况下，旁瓣进一步减小，可使 99.96%的能量集中在主瓣内。

【例 7-7】利用低通滤波器设计海明窗。

其实现的 MATLAB 程序代码如下：

```
>>clear all;  %清除 MATLAB 工作空间中内存变量
wp=0.25*pi;
ws=0.4*pi;
width=ws-wp;
m=ceil(6.6*pi/width)+1;
disp(['滤波器的长度为',num2str(m)]);
n=0:m-1;
%理想 LPF 的截止频率
wc=(ws+wp)/2;
hd=id_lp(wc,m);
w_hm=(hamming(m))';
h=hd.*w_hm;
[db,mag,pha,grd,w]=freqz_m(h,[1]);
de_w=2*pi/1000;
%求出实际通带波动
rp=-(min(db(1:1:wp/de_w+1)));
disp(['实际通带波动为',num2str(rp)]);
%求出最小阻带衰减
as=-round(max(db(ws/de_w+1:1:501)));
disp(['最小阻带衰减为',num2str(as)]);
%绘图
subplot(221),
stem(n,hd);
title('理想冲激响应');
axis([0 m-1 -0.1 0.3]);
ylabel('hd(n)');
subplot(222),
stem(n,w_hm);
title('海明窗');
axis([0 m-1 0 1.1]);
ylabel('w(n)');
subplot(223),
stem(n,h);
title('实际冲激响应');
axis([0 m-1 -0.1 0.3]);
xlabel('n'); ylabel('h(n)');
subplot(224),
plot(w/pi,db);
title('幅度响应(dB)');
```

```
axis([0 1 -100 10]);
grid on;
xlabel('以 pi 为单位的频率'); ylabel('分贝数');
```

运行结果如下：

```
滤波器的长度为 45
实际通带波动为 291.5264
最小阻带衰减为 25
```

同时得到如图 7-12 所示的图形结果。从结果可以看出，滤波器的长度为 $M=67$，实际阻带衰减为 25dB，由此结果可以计算出通带波动为 291.5264dB，这显然是不满足要求的。

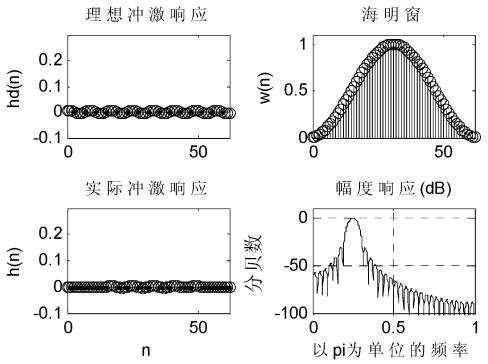


图 7-12　低通滤波器设计图

7.3.5　布莱克曼窗

在 MATLAB 信号处理工具箱中，提供了 bartlett 函数实现布莱克曼窗。其调用格式如下：

w = bartlett(L)

布莱克曼窗的表达式如下：

（1）当 N 为奇数时：

$$w(k)=\begin{cases}\dfrac{2(k-1)}{N-1}, & 1\leqslant k\leqslant\dfrac{N+1}{2}\\ 2-\dfrac{2(k-1)}{N-1}, & \dfrac{N+1}{2}\leqslant k\leqslant N\end{cases}\tag{7-31}$$

（2）当 N 为偶数时：

$$w(k)=\begin{cases}\dfrac{2(k-1)}{N-1}, & 1\leqslant k\leqslant\dfrac{N}{2}\\ 2-\dfrac{2(N-k)}{N-1}, & \dfrac{N}{2}\leqslant k\leqslant N\end{cases}\tag{7-32}$$

布莱克曼窗的频谱的幅度函数为

$$\begin{aligned}W(\omega)=&0.42W_R(\omega)+0.25\left[W_R\left(\omega-\frac{2\pi n}{N-1}\right)+W_R\left(\omega+\frac{2\pi n}{N-1}\right)\right]+\\&0.04\left[W_R\left(\omega-\frac{4\pi n}{N-1}\right)+W_R\left(\omega+\frac{4\pi n}{N-1}\right)\right]\end{aligned}\tag{7-33}$$

【例 7-8】设计布莱克曼窗。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Nwin=20;                                          %数据总数
n=0:Nwin-1;                                       %数据序列序号
w=bartlett(Nwin);                                 %布莱克曼窗
subplot(2,2,1);stem(n,w);                         %绘出窗函数
xlabel('n');ylabel('w(n)');
title('布莱克曼窗');
grid on;
Nf=512;                                           %窗函数复数频率特性的数据点数
Nwin=20;                                          %窗函数数据长度
[y,f]=freqz(w,1,Nf);
mag=abs(y);                                       %求得窗函数幅频特性
w=bartlett(Nwin);                                 %布莱克曼窗;
subplot(2,2,2);plot(f/pi,20*log10(mag/max(mag))); %绘制窗函数的幅频特性
xlabel('归一化频率');ylabel('振幅/dB');
title('bartlett 幅频特性');
grid on;
w=blackman(Nwin);
[y,f]=freqz(w,1,Nf);
mag=abs(y);   %求得窗函数幅频特性
subplot(2,2,3);plot(f/pi,20*log10(mag/max(mag)));   %绘制窗函数的幅频特性
xlabel('归一化频率');ylabel('振幅/dB');
title('blackman 幅频形状');
grid on;
```

运行程序，效果如图 7-13 所示。

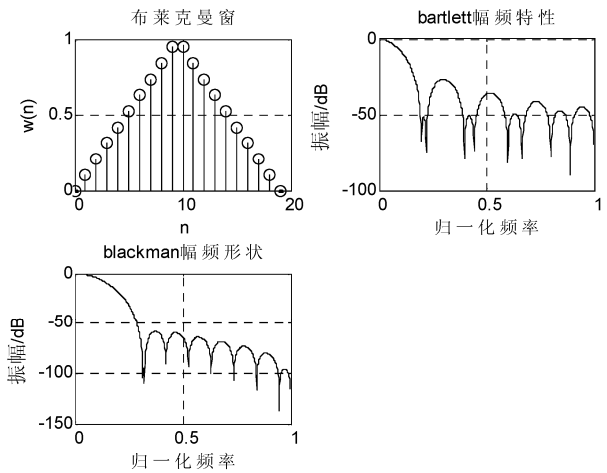


图 7-13　布莱克曼窗

7.3.6　凯赛窗

在 MATLAB 信号处理工具箱中，提供了 kaiser 函数实现凯赛窗，其调用格式如下：

w = kaiser(L,beta)

其中：beta 是 Kaiser 窗参数，影响窗旁瓣幅值的衰减率。

凯赛窗表达式为

$$w(k)=\frac{I_0\left[\beta\sqrt{1-\left(2-\frac{2k}{N-1}\right)^2}\right]}{I_0[\beta]} \tag{7-34}$$

式中：$I_0[\beta]$ 是修正过的零阶贝塞尔函数。

凯赛窗用于滤波器设计时，若旁瓣幅值为 $-\alpha\mathrm{d}\beta$，则

$$\beta=\begin{cases}0.1102(\alpha-8.7), & \alpha>50\\ 0.5842(\alpha-21)^{0.4}+0.07886(\alpha-21), & 50\geqslant\alpha\leqslant21\\ 0, & \alpha<21\end{cases} \tag{7-35}$$

【例 7-9】绘制凯赛窗的频响。

```
>> clear all;
w = kaiser(200,2.5);
wvtool(w)
```

运行程序，效果如图 7-14 所示。

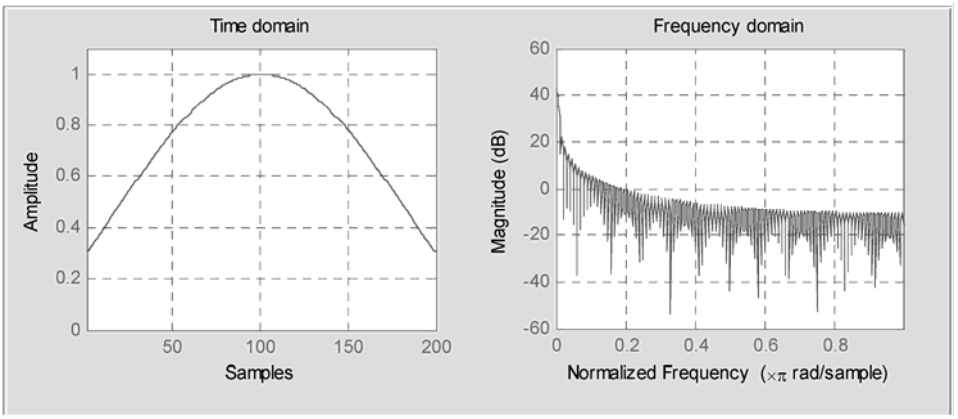


图 7-14　凯赛窗的频响

【例 7-10】利用低通滤波器设计凯赛窗。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=0.2*pi;    ws=0.3*pi; As=50;
tr_width=ws-wp;
M=ceil((As-7.95)/(14.36*tr_width/(2*pi))+1)+1;
n=[0:1:M-1];
beta=0.1102*(As-8.7);
wc=(ws+wp)/2;
hd=ideal_lp(wc,M);
w_kai=(kaiser(M,beta))';
h=hd.*w_kai;
```

```
[db,mag,pha,grd,w]=freqz_m(h,[1]);
delta_w=2*pi/1000;
As=-round(max(db(ws/delta_w+1:1:501)));  %最小阻带衰减
subplot(2,2,1);stem(n,hd);
title('理想脉冲响应');
axis([0 M-1 -0.1 0.3]);
ylabel('hd(n)'); text(M+1,-0.1,'n');
subplot(2,2,2);stem(n,w_kai);
title('凯赛窗');
axis([0 M-1 0 1.1]);
ylabel('w(n)'); text(M+1,0,'n');
subplot(2,2,3);stem(n,h);
title('实际脉冲响应');
axis([0 M-1 -0.1 0.3]);
xlabel('n');ylabel('h(n)');
subplot(2,2,4);plot(w/pi,db);
title('振幅响应');
grid on;
xlabel('频率/pi');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0 0.2 0.3 1]);
set(gca,'YTickMode','manual','YTick',[-50 0]);
set(gca,'YTickLabelMode','manual','YTickLabels',['50','0']);
```

运行程序，效果如图 7-15 所示。

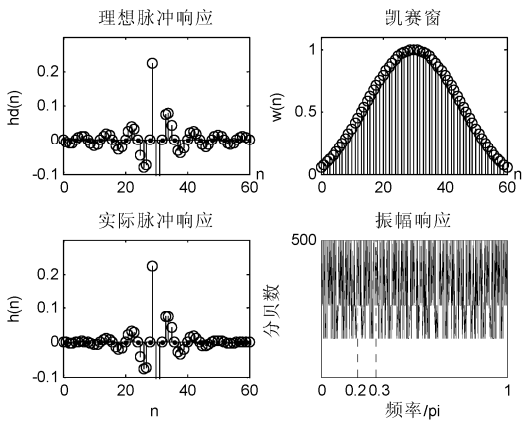


图 7-15　低通滤波器设计凯赛窗

在运行程序过程中，调用了用户自定义编写的 ideal_lp 函数，其源代码如下：

```
function hd=ideal_lp(wc,M)
%  理想低通滤波器计算
% hd 为 0 到 M-1 之间的理想脉冲响应
% wc 为截止频率
% M 为理想滤波器的长度
alpha=(M-1)/2;
n=0:M-1;
m=n-alpha+eps;
hd=sin(wc*n)./(pi*m);
```

7.4 频率取样的 FIR 滤波器设计

工程上，常给定频域上的技术指标，所以采用频域设计更直接。

7.4.1 约束条件

为了设计线性相位的 FIR 滤波器，采样值 $H(k)$ 要满足一定的约束条件。

具有线性相位的 FIR 滤波器，其单位采样响应 $h(n)$ 是实序列，且满足 $h(n)=\pm h(N-1-n)$，由此得到的幅频和相频特性，就是对 $H(k)$ 的约束。

例如，要设计第一类线性相位 FIR 滤波器，即 N 为奇数，$h(n)$ 偶对称，则

$$H(\mathrm{e}^{\mathrm{j}\omega})=H(\omega)\mathrm{e}^{-\mathrm{j}\omega\left(\frac{N-1}{2}\right)} \tag{7-36}$$

幅度函数 $H(\omega)$ 应具有偶对称性：

$$H(\omega)=H(2\pi-\omega) \tag{7-37}$$

因为 $H(\omega)$ 具有偶对称性，所以 $H_k=-H_{N-k}$ 满足对称性（其中 H_k 为任一幅度函数）。

同样，若要设计第二种线性相位 FIR 滤波器，N 为偶数，$h(n)$ 偶对称，相位关系同上，由于幅度特性是奇对称的，即

$$H(\omega)=-H(2\pi-\omega) \tag{7-38}$$

因此，H_k 也必须满足对称要求：

$$H_k=-H_{N-k} \tag{7-39}$$

其他两种线性相位 FIR 数字滤波器的设计，同样也要满足幅度与相位的约束条件。

7.4.2 设计误差

设计步骤：$\theta_k, H_k \Rightarrow H(k) \Rightarrow H(\mathrm{e}^{\mathrm{j}\omega})$

推导：

因

$$\begin{aligned} H(z)&=\sum_{n=0}^{N-1}h(z)z^{-n}=\sum_{n=0}^{N-1}\left[\frac{1}{N}\sum_{k=0}^{N-1}H(k)\mathrm{e}^{\mathrm{j}2\pi k/N}\right]z^{-n} \\ &=\frac{1}{N}\sum_{k=0}^{N-1}H(k)\left[\sum_{k=0}^{N-1}\mathrm{e}^{\mathrm{j}2\pi k/N}z^{-n}\right]=\frac{1}{N}\sum_{k=0}^{N-1}H(k)\frac{1-z^{-N}}{1-\mathrm{e}^{\mathrm{j}2\pi k/N}z^{-1}} \end{aligned} \tag{7-40}$$

令 $W=\mathrm{e}^{\mathrm{j}2\pi k/N}$，则

$$H(z)=\frac{1-z^{-N}}{N}\sum_{k=0}^{N-1}\frac{H(k)}{1-w^{-k}z^{-1}} \tag{7-41}$$

单位圆上的频响为

$$H(\mathrm{e}^{\mathrm{j}\omega})=\frac{1-\mathrm{e}^{-\mathrm{j}\omega N}}{N}\sum_{k=0}^{N-1}\frac{H(k)}{1-\mathrm{e}^{\mathrm{j}2\pi k/N}\mathrm{e}^{-\mathrm{j}\omega}}$$
$$=\frac{1}{N}\sum_{k=0}^{N-1}\frac{H(k)\sin(\omega N/2)}{\sin[(\omega-2\pi kN)/2]}\mathrm{e}^{-\mathrm{j}\left(\frac{N-1}{2}\omega+\frac{k\pi}{N}\right)}=\sum_{k=0}^{N-1}H(k)\phi_k(\mathrm{e}^{\mathrm{j}\omega}) \tag{7-42}$$

式（7-42）是一个内插公式，式中

$$\phi_k(\mathrm{e}^{\mathrm{j}\omega})=\frac{1}{N}\frac{\sin(\omega N/2)}{\sin[(\omega-2\pi kN)/2]}\mathrm{e}^{-\mathrm{j}\left(\frac{N-1}{2}\omega+\frac{k\pi}{N}\right)} \tag{7-43}$$

为内插函数。

令

$$\omega=\frac{2\pi}{N}i,\quad i=0,1,\cdots,N-1 \tag{7-44}$$

则

$$\phi_k(\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}i})=\begin{cases}1, & k=i\\ 0, & k\neq i\end{cases}\quad i=1,2,\cdots,N-1 \tag{7-45}$$

【例 7-11】频率采样技术：低通，最优法 T1 & T2。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=0.2*pi; ws=0.3*pi;
Rp=0.25; Rs=50;
T1=0.5925;     Ts=0.1099;
M=60;   alpha=(M-1)/2;   l=0:M-1; w1=(2*pi/M)*l;
Hrs=[ones(1,7),T1,0.11,zeros(1,43),0.11,T1,ones(1,6)];
Hdr=[1 1 0 0];wdl=[0 0.2 0.3 1];
k1=0:floor((M-1)/2);k2=floor((M-1)/2)+1:M-1;
angH=[-alpha*(2*pi)/M*k1,alpha*(2*pi)/M*(M-k2)];
H=Hrs.*exp(j*angH);
h=real(ifft(H,M));
[db,mag,pha,grd,w]=freqz_m(h,1);
[Hr,ww,a,L]=hr_type2(h);
subplot(2,2,1);plot(w1(1:31)/pi,Hrs(1:31),'o',wdl,Hdr);
axis([0,1,-0.1,1.1]);title('低通:M=60, T1=0.59, T2=0.109');
xlabel(''); ylabel('Hr(k)');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.3,1]);
set(gca,'YTickMode','manual','YTick',[0,0.059,0.109,1]);
grid on;
subplot(2,2,2);stem(l,h);axis([-1,M,-0.1,0.3]);
title('脉冲响应');ylabel('h(n)');text(M+1,-0.1,'n');
subplot(2,2,3);plot(ww/pi,Hr,w1(1:31)/pi,Hrs(1:31),'o');
axis([0,1,-0.1,1.1]);title('振幅响应');
xlabel('频率/pi');ylabel('Hr(w)');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.3,1]);
```

```
set(gca,'YTickMode','manual','YTick',[0,0.059,0.109,1]);
grid on;
subplot(2,2,4);plot(w/pi,db);
axis([0 1 -100 10]);
grid on;title('幅度响应');
xlabel('频率/pi');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.3,1]);
set(gca,'YTickMode','manual','YTick',[-63;0]);
set(gca,'YTickLabelMode','manual','YTickLabels',['63';' 0']);
```

运行程序，效果如图 7-16 所示。

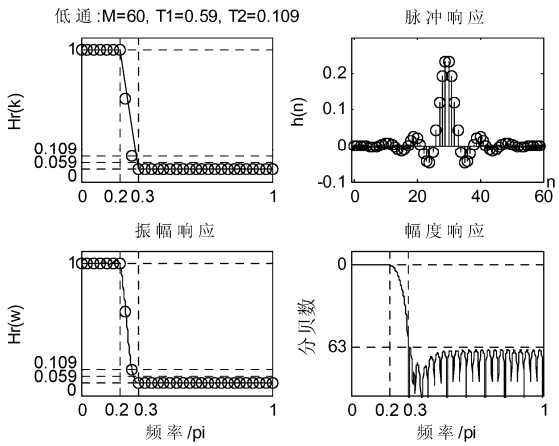


图 7-16　频率采样技术：低通，最优法 T1 & T2

【例 7-12】频率采样技术：高通，最优法 T1。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp=0.8*pi; ws=0.6*pi;
Rp=1; As=50;
T1=0.1095;     T2=0.598;
M=33;    alpha=(M-1)/2;    l=0:M-1; w1=(2*pi/M)*l;
Hrs=[zeros(1,11),T1,T2,ones(1,8),T2,T1,zeros(1,10)];
Hdr=[0 0 1 1];wdl=[0 0.6 0.8 1];
k1=0:floor((M-1)/2);k2=floor((M-1)/2)+1:M-1;
angH=[-alpha*(2*pi)/M*k1,alpha*(2*pi)/M*(M-k2)];
H=Hrs.*exp(j*angH);
h=real(ifft(H,M));
[db,mag,pha,grd,w]=freqz_m(h,1);
[Hr,ww,a,L]=hr_type1(h);
subplot(1,1,1)
subplot(2,2,1);plot(w1(1:17)/pi,Hrs(1:17),'o',wdl,Hdr);
axis([0,1,-0.1,1.1]);title('高通:M=33, T1=0.1095, T2=0.598');
xlabel(''); ylabel('Hr(k)');
set(gca,'XTickMode','manual','XTick',[0;.6;.8;1]);
set(gca,'XTickLabelMode','manual','XTickLabels',[' 0';'.6';'.8';' 1']);
set(gca,'YTickMode','manual','YTick',[0,0.109,0.59,1]);
```

```
grid on;
subplot(2,2,2);stem(l,h);axis([-1,M,-0.4,0.4]);
title('脉冲响应');ylabel('h(n)');text(M+1,-0.4,'n');
subplot(2,2,3);plot(ww/pi,Hr,w1(1:17)/pi,Hrs(1:17),'o');
axis([0,1,-0.1,1.1]);title('振幅响应');
xlabel('频率/pi');ylabel('Hr(w)');
set(gca,'XTickMode','manual','XTick',[0,.6,.8,1]);
set(gca,'XTickLabelMode','manual','XTickLabels',[' 0';'.6';'.8';' 1']);
set(gca,'YTickMode','manual','YTick',[0,0.109,0.59,1]);
grid on;
subplot(2,2,4);plot(w/pi,db);
axis([0 1 -100 10]);
grid on;title('幅度响应');
xlabel('频率/pi');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0;.6;.8;1]);
set(gca,'XTickLabelMode','manual','XTickLabels',[' 0';'.6';'.8';' 1']);
set(gca,'YTickMode','manual','YTick',[-50;0]);
set(gca,'YTickLabelMode','manual','YTickLabels',['50';' 0']);
```

运行程序，效果如图 7-17 所示。

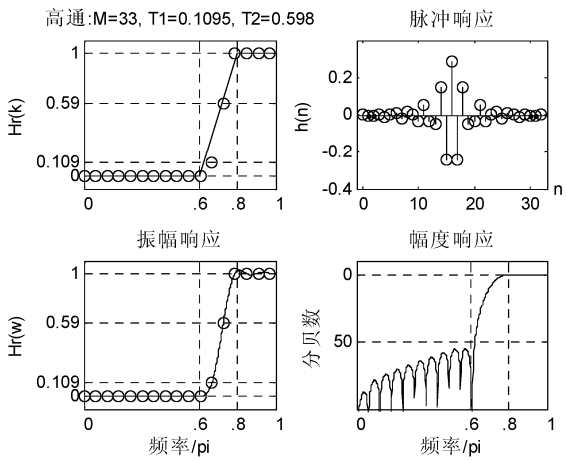


图 7-17　频率采样技术：高通，最优法 T1

【例 7-13】频率采样技术：带通，最优法 T1&T2。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
wp1=0.35*pi; ws1=0.2*pi;
wp2=0.65*pi; ws2=0.8*pi;
Rp=1;   Rs=60;
T1=0.109021;     T2=0.59417456;
M=40;   alpha=(M-1)/2;   l=0:M-1; w1=(2*pi/M)*l;
Hrs=[ones(1,5),T1,T2,zeros(1,7),T2,T1,ones(1,9),T1,T2,ones(1,7),T2,T1,zeros(1,4)];
Hdr=[0 0 1 1 0 0];wdl=[0 0.2 0.35 0.65 0.8 1];
k1=0:floor((M-1)/2);k2=floor((M-1)/2)+1:M-1;
angH=[-alpha*(2*pi)/M*k1,alpha*(2*pi)/M*(M-k2)];
H=Hrs.*exp(j*angH);
```

```
h=real(ifft(H,M));
[db,mag,pha,grd,w]=freqz_m(h,1);
[Hr,ww,a,L]=hr_type2(h);
subplot(2,2,1);plot(w1(1:21)/pi,Hrs(1:21),'o',wdl,Hdr);
axis([0,1,-0.1,1.1]);title('低通:M=40, T1=0.5941, T2=0.109');
xlabel(''); ylabel('Hr(k)');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.35,0.65,0.8,1]);
set(gca,'YTickMode','manual','YTick',[0,0.059,0.109,1]);
grid on;
subplot(2,2,2);stem(l,h);axis([-1,M,-0.4,0.4]);
title('脉冲响应');ylabel('h(n)');text(M+1,-0.4,'n');
subplot(2,2,3);plot(ww/pi,Hr,w1(1:21)/pi,Hrs(1:21),'o');
axis([0,1,-0.1,1.1]);title('振幅响应');
xlabel('频率/pi');ylabel('Hr(w)');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.35,0.65,0.8,1]);
set(gca,'YTickMode','manual','YTick',[0,0.059,0.109,1]);
grid on;
subplot(2,2,4);plot(w/pi,db);
axis([0 1 -100 10]);
grid on;title('幅度响应');
xlabel('频率/pi');ylabel('分贝数');
set(gca,'XTickMode','manual','XTick',[0,0.2,0.35,0.65,0.8,1]);
set(gca,'YTickMode','manual','YTick',[-60;0]);
set(gca,'YTickLabelMode','manual','YTickLabels',['60';' 0']);
```

运行程序，输出如图 7-18 所示。

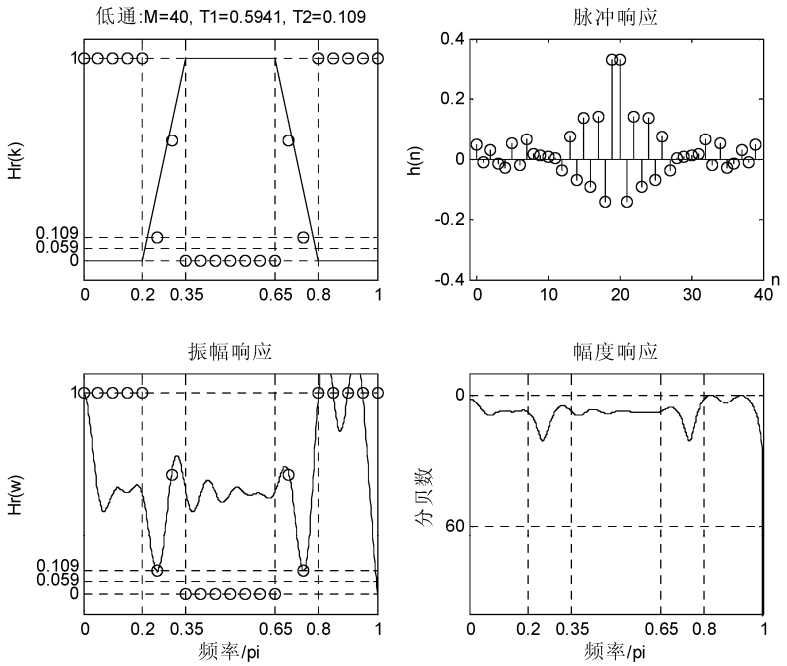


图 7-18　频率采样技术：带通，最优法 T1&T2

7.5　最优的 FIR 滤波器设计

MATLAB 信号处理工具箱提供了比基于窗函数法 FIR 滤波器设计工具箱函数 fir1 和 fir2 更为通用的函数 firl2 和 remez。它们采用不同的优化方法设计最优的标准多频带 FIR 数字滤波器。

函数 remez 实现 Park-McClellan 算法，这种算法利用 Remez 交换算法和 Chebyshev 近似理论来设计滤波器，使实际频率响应拟合期望频率响应达到最优。从实际和理想频响之间最大误差最小化的观点来看，函数 remez 设计的滤波器是最优的，因此，又称之为最优滤波器。在频率域内，滤波器呈现等波纹特点，因此，又称之为等波纹器。Park-McClellan 滤波器设计方法是 FIR 滤波器设计中最流行的，应用最广的设计方法。

函数 firls 和 remez 的调用格式相同，只是优化算法不同。

7.5.1　一般最优滤波器

firls 和 remez 的基本调用格式如下：

hd = firls(d)

b=remez(n, f, a)

式中：n 为滤波器阶数；f 为滤波器期望频率特性归一化频率向量，范围为 0～1，为递增向量，允许定义重复频率点；a 为滤波器期望频率特性的幅值向量，向量 a 和 f 必须为同长度，且为偶数；b 为返回滤波器系数，长度为 n+1，且具有偶对称的关系，即 b(k)=b(n+2-k)。若滤波器的阶数为奇数，则在奈奎斯特频率处（对应于归一化频率 1），幅频响应必须为 0。滤波器的阶数为偶数则无此限制。

函数 firls 和 remez 可用于设计低通、高通、带通和带阻等一般类型的滤波器，这可由函数中给定的理想幅频响应的频率向量 f 和幅值向量确定。

如设计一个带通滤波器，幅频响应定义为

$$f = [0\ 0.3\ 0.4\ 0.7\ 0.8\ 1],\quad a = [0\ 0\ 1\ 1\ 0\ 0]$$

则该理想滤波器幅频响应定义为：阻带频率 0～0.3，0.8～1，通带频率 0.4～0.7，过渡带 0.3～0.4，0.7～0.8。

设计一个高通滤波器，如果想幅频响应向量对的给定形式为：$f = [0\ 0.7\ 0.8\ 1]$，$a = [0\ 0\ 1\ 1]$，则该理想滤波器的幅频响应定义为：阻带频率 0～0.7，通带频率 0.8～1.0，过渡带 0.7～0.8。

设计一个带阻滤波器，如果想幅频响应向量对的给定形式为：$f = [0\ 0.3\ 0.4\ 0.5\ 0.6\ 1]$，$a = [1\ 1\ 0\ 0\ 1\ 1]$，则该阻带频率 0.4～0.5，通带频率 0～0.3，0.6～1.0，过渡带 0.3～0.4，0.5～0.6。

此外，函数 firls 和 remez 还可以设计多带滤波器。

【例 7-14】用函数 firls 和 remez 设计一个 50 阶多通滤波器，滤波器理想频率响应对为：f=[0 0.1 0.15 0.25 0.3 0.4 0.45 0.55 0.6 0.7 0.75 0.85 0.9 1]，a=[1 1 0 0 1 1 0 0 1 1 0 0 1 1]；将设计的滤波器的幅频响应和理想滤波器幅频响应进行比较，绘制 remez 函数设计滤波器的脉冲响应。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=50;                    %滤波器的阶数
f=[0 0.1 0.15 0.25 0.3 0.4 0.45 0.55 0.6 0.7 0.75 0.85 0.9 1];   %频率向量
a=[1 1 0 0 1 1 0 0 1 1 0 0 1 1];              %振幅向量
```

```
b=firls(n,f,a);          %采用 firls 设计滤波器
[h,w1]=freqz(b);         %计算滤波器的频率响应
bb=remez(n,f,a);         %采用 remez 设计滤波器
[hh,w2]=freqz(bb);       %计算滤波器的频率响应
figure;
plot(w1/pi,abs(h),'r.',w2/pi,abs(hh),'b-.',f,a,'ms');
%绘制滤波器幅频响应
xlabel('归一化频率');ylabel('振幅');
legend('firls 设计滤波器','remez 设计滤波器','理想特性');
figure;
impz(bb,1),title('脉冲响应');   %给出滤波器的脉冲响应
xlabel('样本数');ylabel('幅度');
```

运行程序，效果如图 7-19 与图 7-20 所示。

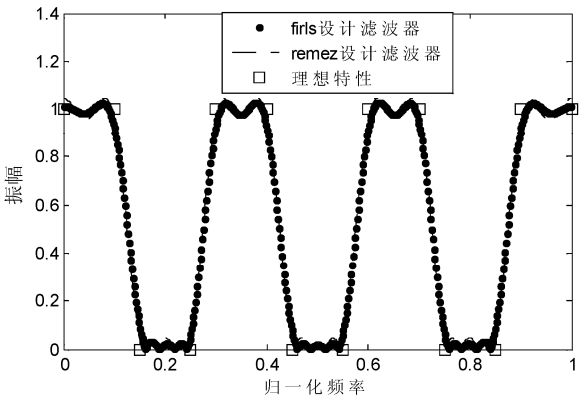


图 7-19　设计滤波器的幅频响应与理想滤波器幅频响应的比较

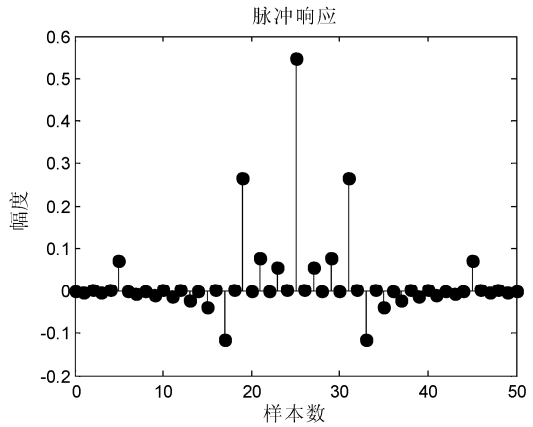


图 7-20　设计滤波器的脉冲响应

由图可以看出，firls 所设计的滤波器的通带和阻带具有较小的波纹，但在整个频带内不一致。而 remez 函数设计的滤波器具有较大的通带和阻带波纹，但在整个频带内较为一致。图 7-20 给出了 remez 函数设计滤波器的脉冲响应，且具有偶对称的关系，即 b(k)=b(n+2-k)。大家还可以观看 firls 函数设计滤波器的脉冲响应是否具有偶对称关系。

函数 firls 和 remez 还能设计具有任意线性过渡带连接阻带和通带，使过渡带具有更广阔平滑的过渡区间。如理想幅频响应按如下频响对给出：f=[0 0.4 0.42 0.48 0.5 1]，a=[1 1 0.8 0.2 0 0];

这里，过渡带 0.4～0.5 给出多个响应值设计出的具有线性过渡带 FIR 滤波器的频率响应。

7.5.2　加权最优滤波器

函数 firls 和 remez 还可以增加输入参数设置权向量 w。在不同频段设置不同权值，使不同频段的误差值最小化得到不同程度的重视。具有权向量输入的函数 firls 和 remez 实现滤波器的每个频率段加权处理。其调用格式如下：

b=firls(n, f, a, w)

b=remez(n, f, a ,w)

式中：w 为权向量，为 f 和 a 向量长度的一半，一个频带必须对应一个权值。

【例 7-15】设计一个 30 阶的低通等波纹的滤波器，通常边界频率 0.4，阻带边界频率 0.5，均为归一化频率，阻带波纹约为通带波纹的 1/10。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=30;                  %滤波器的阶数
f=[0 0.4 0.5 1];       %频率向量
a=[1 1 0 0];           %振幅向量
w=[1 10];
b=firls(n,f,a,w);      %采用 firls 设计滤波器
[h,w1]=freqz(b);       %计算滤波器的频率响应
bb=remez(n,f,a,w);     %采用 remez 设计滤波器
[hh,w2]=freqz(bb);     %计算滤波器的频率响应
figure;
plot(w1/pi,abs(h),'r.',w2/pi,abs(hh),'b-.',f,a,'ms');
%绘制滤波器幅频响应
xlabel('归一化频率');ylabel('振幅');
```

运行程序，效果如图 7-21 所示。

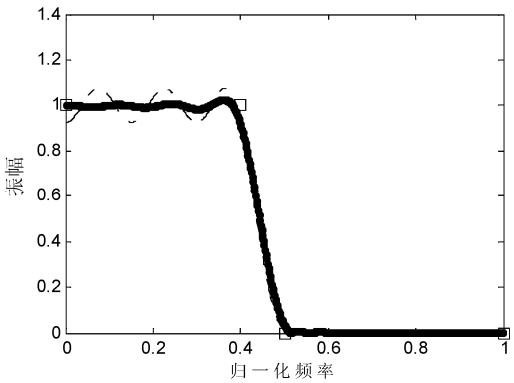


图 7-21　设计滤波器的幅频响应

由图可以看出，该滤波器对阻带和通带波纹进行了控制，得到了符合要求的滤波器。由于 remez 要求等波纹的特点，因此，其在通带内的振动振幅较大，在阻带内的振动振幅确实较小，约为通带内振动振幅的 1/10，符合设计要求。如果滤波器的阶数足够高，则可以获得更为理想的幅频响应。

7.5.3 反对称 FIR 滤波器

利用函数 firls 和 remez 可设计滤波器系数为奇对称的滤波器，即

$$b(n) = -b(N-1-n) \tag{7-46}$$

这类滤波器要求滤波器的零频响应为 0，若滤波器阶数为偶数，则还要求奈奎斯特频率（归一频率为 1）处的频率响应为零。该滤波器除了能对信号保持线性相位滤波外，还能实现信号的赫尔伯特变换，故又称赫尔伯特变换器。所谓赫尔伯特变换就是使信号通过赫尔伯特变换后负频率作+90° 的相移，正频率作−90° 的相移，而振幅不发生改变。在地震学研究中，地震射线通过焦散而后会发生 90° 相移的畸变，通常采用赫尔伯特变换进行校正来正确识别地震震相。

函数调用格式如下：

```
b=firls(n ,f, a, 'h')
b=filrs(n, f, a, w, 'h')
```

或

```
b=remez(n, f, a, 'h')
b=remez(n, f, a, w, 'h')
```

其中：'h'为选择项，表示设计的滤波器是奇对称线性相位滤波器。

【例 7-16】利用函数 remez 和 firls 设计一个高通反对称线性相位滤波器，并绘制其频率特征图。设计一个采样频率为 50Hz，频率为 5Hz 的振动作为输入信号，试验对应的数据点是否满足相位相差 90° 的特点。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=21;                          %滤波器的阶数
f=[0 0.1 0.2 1];               %频率向量
a=[0 0 1 1];                   %振幅向量
b=firls(n,f,a,'h');            %采用 firls 设计奇对称系数滤波器
[h,w1]=freqz(b,1,512);         %计算滤波器的频率响应
bb=remez(n,f,a,'h');           %采用 remez 设计奇对称系数滤波器
[hh,w2]=freqz(bb,1,512);       %计算滤波器的频率响应
figure;
plot(w1/pi,abs(h),'r.',w2/pi,abs(hh),'b-.',f,a,'ms');
%绘制滤波器幅频响应
xlabel('归一化频率');ylabel('振幅');
legend('firls 设计滤波器','remez 设计滤波器','理想特性');
t=0:1/50:3;                    %时间序列
x=sin(2*pi*5*t);               %输入信号
figure;
subplot(2,1,1);plot(t(1:100),x(1:100));
%绘制输入信号的前 100 个样本
title('输入信号');
y1=filter(b,1,x);              %运用 firls 设计的滤波器进行滤波
y2=filter(bb,1,x);             %运用 remez 设计的滤波器进行滤波
subplot(2,1,2);plot(t([1:100]+20/2),y1([1:100]+20/2),t([1:100]+20/2),y2([1:100]+20/2),'r:');
%绘制与输入信号对应的输出信号,考虑了延迟效应
legend('firls 滤波器','remez 滤波器');
```

```
xlabel('时间/s');title('输出信号');
```

运行程序，输出如图 7-22 及图 7-23 所示。

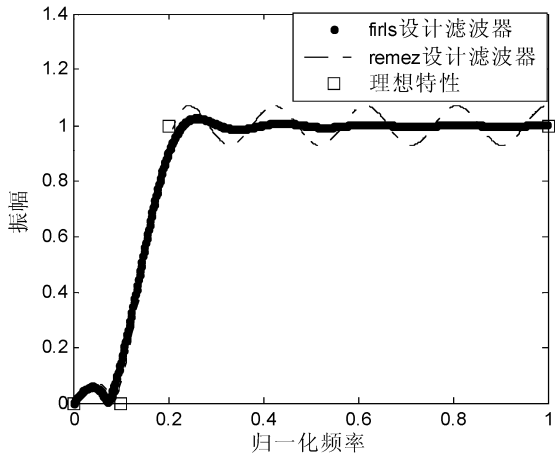


图 7-22　设计滤波器的幅频响应

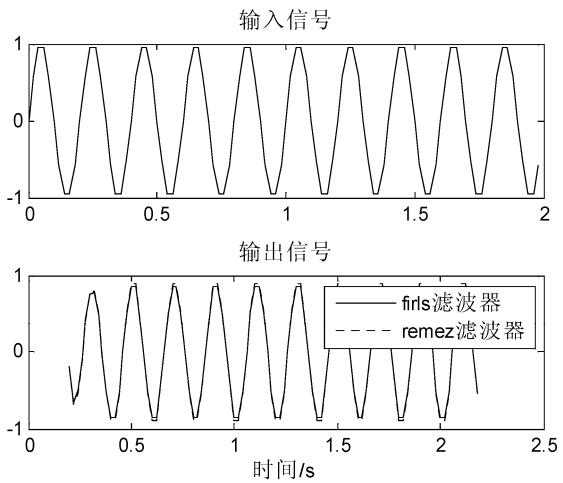


图 7-23　设计滤波器的输入信号及输出信号

由图可以看出，该滤波器对阻带和通通波纹进行了控制，得到了符合要求的滤波器。由于两种滤波器的幅频响应非常一致，因此采用两种滤波器的输出信号的振幅也很接近。但是跟输入信号比较可知，输出信号比输入信号前移了 90° 的相位，这正是赫尔伯特变换的结果。如果将输入信号改为−5Hz，可以看到输出的信号后移 90° 的相位。读者可以动手修改程序，比较其结果。

7.5.4　微分 FIR 滤波器

对信号时间域的微分等价于信号的傅里叶变换和一个虚数单位斜坡函数的乘积。也就是说，信号的微分相当于让该信号通过一个频率响应为 ω 的滤波器。函数 firls 和 remez 可用于设计这种具有微分作用的 FIR 滤波器，调用格式如下：

```
b=firls(n, f, a, 'd')
b=remez(n, f, a, 'd')
```

其中：'d'是选项项，表示设计的滤波器具有微分器的作用。

【例 7-17】利用 remez 和 firls 设计一个 FIR 微分器，频率在 0～0.9 范围内，绘制其频率特性图。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=30;                          %滤波器的阶数
f=[0 0.9];                     %频率向量
a=[0 0.9];                     %振幅向量
b=firls(n,f,a,'d');            %采用 firls 设计滤波器
[h,w1]=freqz(b);               %计算滤波器的频率响应
bb=remez(n,f,a,'d');           %采用 remez 设计滤波器
[hh,w2]=freqz(bb);             %计算滤波器的频率响应
figure;
plot(w1/pi,abs(h),'r.',w2/pi,abs(hh),'b-.',f,a,'ms');
%绘制滤波器幅频响应
xlabel('归一化频率');ylabel('振幅');
legend('firls 设计滤波器','remez 设计滤波器','理想特性',4);
%绘制图例,4 表示图例的位置在右下角
grid on;
f1=5;                          %输入信号频率
t=0:1/1000:1;                  %时间序列
x=sin(2*pi*f1*t);              %输入信号
y=fftfilt(bb,x);               %滤波
figure;
subplot(2,1,1);plot(t,x);      %绘制输入信号
title('输入信号');
subplot(2,1,2);plot(t,y);      %绘制输出信号
ylim([-0.01 0.01]);
xlabel('时间/s');title('输出信号');
```

运行程序，效果如图 7-24 及图 7-25 所示。

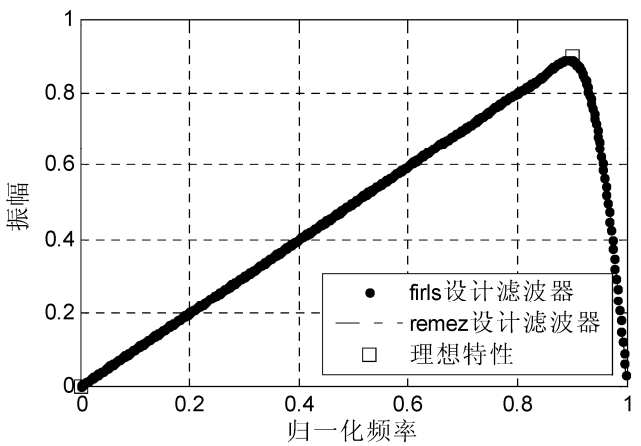


图 7-24　设计滤波器的频率响应

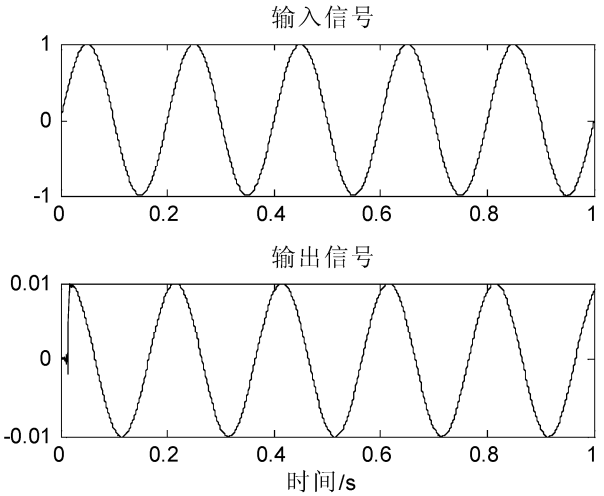


图 7-25　输入信号和输出信号

7.6　IIR 与 FIR 数字滤波器的比较

在第 6 章和第 7 章分别讨论了 IIR 和 FIR 滤波器的各种设计方法，这两种滤波器究竟有什么特点呢？首先，从性能上讲，IIR 滤波器传递函数的极点可位于单位圆内的任何地方，因此可以用较低的滤波器阶数获得高的选择性，所以存储单元少，经济且效率高。但是，IIR 滤波器是以相位的非线性为代价的，滤波器选择性越好，则相位非线性越严重。

相反，FIR 滤波器可以得到严格的线性相位。然而，FIR 滤波器传递函数的极点固定在原点，只能用较高的阶数才能使滤波器达到高的选择性。在相同的性能指标下，FIR 滤波器的阶数要比 IIR 高 5 倍～10 倍，这会造成硬件实现成本高，信号延时较大的问题；如果按相同的选择性和相同的线性要求来说，则 IIR 滤波器就必须加全通网络进行相位校正，同样加大了滤波器的阶数和复杂性。

从结构上看，IIR 滤波器必须采用递归结构，为保证系统的稳定性，极点必须位于单位圆内。实际硬件实现时，这种结构需对序列进行舍入处理，从而会导致产生振荡。相反，FIR 滤波器采用非递归结构，不存在不稳定问题，误差也小。此外，FIR 滤波器还可以采用快速傅里叶变换算法，使运算速度加快。

另外，IIR 滤波器虽然设计简单，但主要是用于设计具有片段常数特性的滤波器，如低通、高通、带通和带阻等滤波器。而 FIR 滤波器则灵活得多，尤其是能适应某些特殊应用。例如，某些情况下，需要求三角形振幅响应或者更复杂的幅频响应，因而应用空间广阔。

总之，FIR 和 IIR 滤波器各有所长，在实际应用中应从多方面加以考虑选择。

第 8 章　其他滤波器

8.1　自适应滤波器

自适应滤波器与维纳滤波器一样都是以最小均方误差为准则的最佳滤波器。自适应滤波器能自动调节其本身的单位脉冲响应 $h(t)$特性，以达到最优化的滤波效果。设计自适应滤波器时，可以不必要求预先知道信号与噪声的自相关函数，而且在滤波过程中，如果信号与噪声的自相关函数即使随时间缓慢变化，系统也能自动适应，自动调节参数，使均方误差最小。

8.1.1　自适应滤波器设计原理

前面对 FIR 滤波器的结构和实现方法已经作了介绍，而自适应滤波器实现的一种常用方法就是使用横向结构 FIR 滤波器。图 8-1 给出了自适应滤波器的结构图。

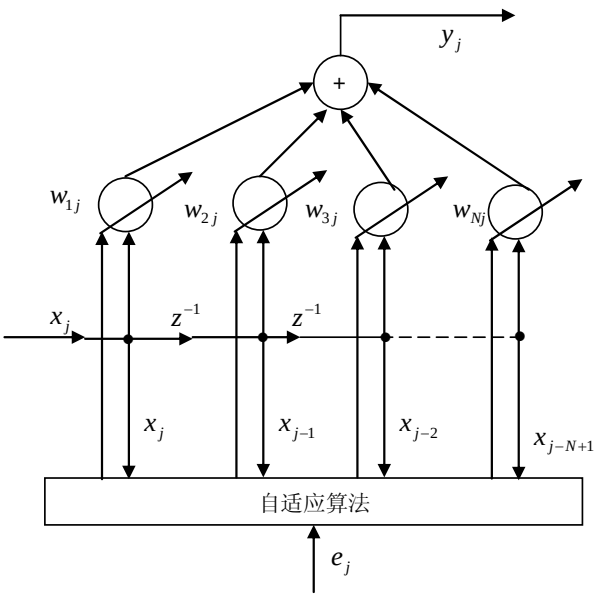


图 8-1　自适应滤波器结构

由维纳-霍夫方程可知，最小均方误差为

$$(E[\boldsymbol{e}_j^2])_{\min} = E[\boldsymbol{d}_j^2] - \boldsymbol{W}^{*\tau}\boldsymbol{P} \tag{8-1}$$

实际上，该方程与维纳滤波器结果完全相同。自适应滤波器与维纳滤波器相比，其差别在于它增加了一个识别控制环节，将输出 $\boldsymbol{y}_j$ 与所希望的值 $\boldsymbol{d}_j$ 进行比较，利用误差 $\boldsymbol{e}_j$ 去控制 $\boldsymbol{W}$，使 $E[\boldsymbol{e}_j^2]$=达到最小值，从而得到 $\boldsymbol{W}$ 的估计 $\boldsymbol{W}^*$。因此，自适应滤波的关键在于如何简便计算 $\boldsymbol{W}^*$。计算 $\boldsymbol{W}^*$ 的常用方法是最小均方算法，简称 LMS 算法。

根据最优化的数学算法最陡下降法，下一个权矢量 $\boldsymbol{W}_{j+1}$ 等于现在的权矢量 $\boldsymbol{W}_j$ 加一个正比于梯度 Δ_j 的负值变化量，即有

$$\boldsymbol{W}_{j+1}=\boldsymbol{W}_j-\mu\boldsymbol{\Delta}_j \tag{8-2}$$

通过梯度下降法：

$$\boldsymbol{W}_{j+1}=\boldsymbol{W}_j-\mu\frac{\mathrm{d}E[\boldsymbol{e}_j^2]}{\mathrm{d}\boldsymbol{W}}\bigg|_{\boldsymbol{W}=\boldsymbol{W}_j} \tag{8-3}$$

推导可得

$$\boldsymbol{W}_{j+1}=\boldsymbol{W}_j+2\mu\boldsymbol{e}_j\boldsymbol{X}_j \tag{8-4}$$

其中：$\boldsymbol{e}_j=\boldsymbol{d}_j-\boldsymbol{W}_j^T\boldsymbol{X}_j$，$\mu$是一个控制稳定性和收敛速度的参数，这里$\mu$>0。

应用 LMS 算法的自适应横向滤波器如图 8-2 所示。

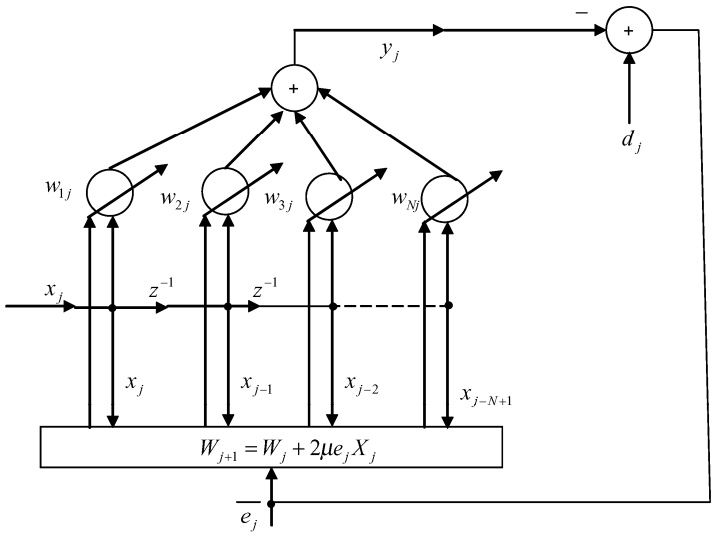


图 8-2　应用 LMS 算法的自适应横向滤波器

8.1.2　自适应滤波器在 MATLAB 中的应用

自适应算法在工程应用中有着广泛的应用，下面通过例子来说明自适应滤波器的设计过程和 MATLAB 实现的代码。

【例 8-1】设计一个 2 阶加权自适应横向滤波器，对一正弦加噪声信号进行滤波。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0:1/10000:1-0.0001;
s=sin(2*pi*t);              %标准的正弦信号
n=randn(size(t));           %产生随机噪声
x=s+n;
w=[0 0.5];
u=0.00026;
for i=1:9999
    y(i+1)=n(i:i+1)*w';
    e(i+1)=x(i+1)-y(i+1);
    w=w+2*u*e(i+1)*n(i:i+1);
end
```

```
figure;
subplot(3,1,1); plot(t,x);
title('带噪声正弦信号');
subplot(3,1,2);plot(t,s);
title('正弦信号');
subplot(3,1,3);plot(t,e);
title('滤波结果');
```

运行程序，效果如图 8-3 所示。

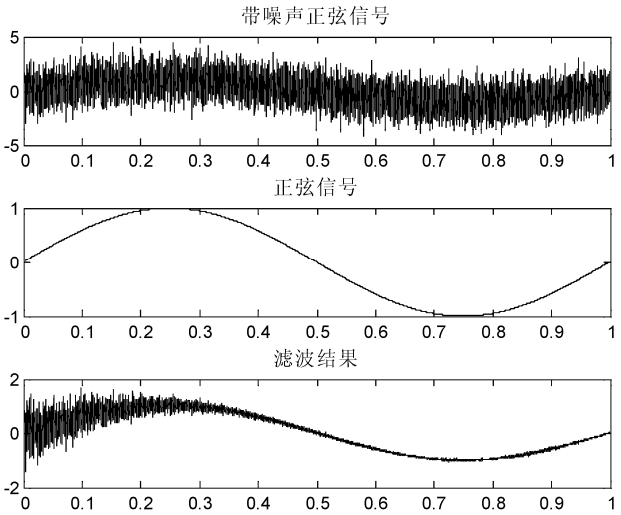


图 8-3　自适应滤波器对噪声信号滤波

利用自适应滤波器，不仅可以实现对信号噪声的自适应滤除，还能用于系统识别。

【例 8-2】通过自适应 FIR 滤波器，识别某未知系统。系统示意如图 8-4 所示。

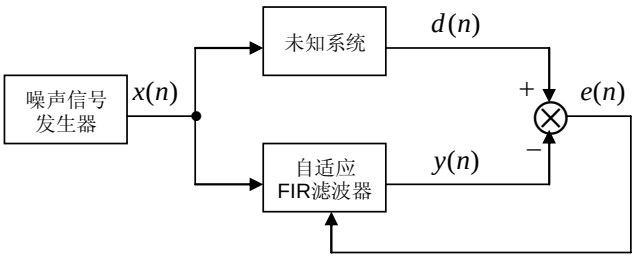


图 8-4　某信号处理系统

以下程序代码利用 FIR 滤波器的自适应调整，不断修正其系统函数，使其与未知系统的参数充分逼近，从而误差最小，达到系统识别的目的。其实现的 MATLAB 程序代码如下：

```
>> clear all;
ee=0;
fs=800;                %采样频率 800Hz
det=1/fs;
f1=100;   f2=200;
t=0:det:2-det;
s=cos(2*pi*f1*t)+cos(2*pi*f2*t);
x=randn(size(t));
```

```
x=s+x;
%  未知系统
[b,a]=butter(5,150*2/fs);          %截止频率取 150Hz
d=filter(b,a,x);                   %自适应 FIR 滤波器
N=5;
delta=0.06;
M=length(x);
y=zeros(1,M);
h=zeros(1,N);
for n=N:M
      x1=x(n:-1:n-N+1);
      y(n)=h*x1';
      e(n)=d(n)-y(n);
      h=h+delta.*e(n).*x1;
end
x=abs(fft(x,2048));
Nx=length(x);
kx=0:800/Nx:(Nx/2-1)*(800/Nx);
D=abs(fft(d,2048));
Nd=length(D);
kd=0:800/Nd:(Nd/2-1)*(800/Nd);
y=abs(fft(y,2048));
Ny=length(y);
ky=0:800/Ny:(Ny/2-1)*(800/Ny);
figure;
subplot(3,1,1);plot(kx,x(1:Nx/2));
xlabel('Hz');title('原始信号频谱');
subplot(3,1,2);plot(kd,D(1:Nd/2));
title('经未知系统后信号频谱');xlabel('Hz');
subplot(3,1,3);plot(ky,y(1:Ny/2));
title('经自适应 FIR 滤波器后信号频谱');xlabel('Hz');
```

运行程序，效果如图 8-5 所示。

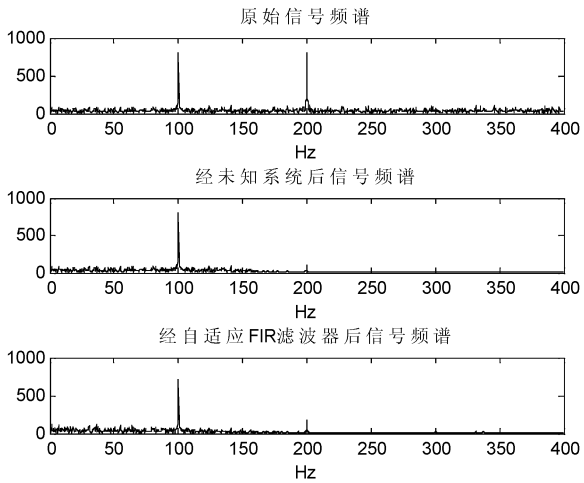


图 8-5　系统信号处理频谱

从图 8-5 可知，自适应 FIR 滤波器能很好地模拟未知系统，它们对原始信号处理后的效果十分接近。这样，通过自适应 FIR 滤波器的参数指标，就能得到未知系统的系统函数，从而可以对未知系统进行功能相同的硬件重构。这在工程应用中有着广泛的应用。

8.2 格型滤波器

1973 年，Gay 和 Markel 提出了一种新的系统结构形式，即 Lattice 结构（又称格型结构）。格型滤波器广泛应用于数字语音处理和自适应滤波器实现中，在这些应用中，它比其他 FIR 或 IIR 滤波器结构更优越，这是因为在语音分析和语音合成中，它可用较少数目的系统为较大数目的谐振峰实时建模。FIR 滤波器的格型滤波器描述是全零点格型，而 IIR 滤波器的描述是格型梯形结构。

8.2.1 全零点格型滤波器

长度为 M（或阶数为 M–1）的 FIR 滤波器具有 M–1 级格型结构，每一级的输入输出通过顺序递归方程建立联系：

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + K_m g_{m-1}(n-1), \quad m = 1,2,\cdots,M-1 \\ g_m(n) &= K_m f_{m-1}(n) + g_{m-1}(n-1), \quad m = 1,2,\cdots,M-1 \end{aligned} \tag{8-5}$$

其中：参数 K_m，$m = 1,2,\cdots,M-1$ 叫做反射系数，它们是格型滤波器的系数。如果 $f_m(n)$ 和 $g_m(n)$ 的初值均为滤波器输出 $x(n)$ 的倍数（乘以 K_0），则 M–1 级格型滤波器的输出与（M–1）阶 FIR 滤波器输出一致，也就是

$$\begin{aligned} f_0(n) &= g(n) = K_0 x(n) \\ y(n) &= f_{M-1}(n) \end{aligned} \tag{8-6}$$

如果 FIR 滤波器以直接形式给出：

$$H(z) = \sum_{m=0}^{M-1} b_m z^{-m} = b_0 \left(1 + \sum_{m=1}^{M-1} \frac{b_m}{b_0} z^{-m} \right) \tag{8-7}$$

同时，把多项式 $A_{M-1}(x)$ 表示为

$$\begin{aligned} A_{M-1}(z) &= \left(1 + \sum_{m=1}^{M-1} a_m(m) z^{-m} \right) \\ a_{M-1}(m) &= \frac{b_m}{b_0}, \quad m = 1,2,\cdots,M-1 \end{aligned} \tag{8-8}$$

那么格型滤波器系数 $\{K_m\}$ 可由下面的递归算法得到：

$$
\begin{aligned}
&K_0 = b_0 \\
&K_{M-1} = a_{M-1}(M-1) \\
&J_m(z) = z^{-m}A_m(z^{-1}), \quad m = M-1,\cdots,1 \\
&A_{M-1}(z) = \frac{A_m(z) - K_m J_m(z)}{1-K_m^2}, \quad m = M-1,\cdots,1 \\
&K_m = a_m(m), \quad m = M-1,\cdots,1
\end{aligned}
\tag{8-9}
$$

在 MATLAB 中，给定直接形式的系数$\{b_n\}$，通过上式可得到格型滤波器系数$\{K_m\}$。这可由下面的 MATLAB 函数 dir2latc 完成。注意计算$J_m(z)$的方程暗示了多项式$J_m(z)$是对多项式$A_m(z)$进行 fliplr 运算而得到的。dir2latc 函数的源代码如下：

```
function [K]=dir2latc(b)
% FIR 的直接形式到全零格型的型式转换
% K 为格型滤波器系数(反射系数);b 为直接形式多项式系数(脉冲响应)
M=length(b);
K=zeros(1,M);
b1=b(1);
if b1==0
    error('b(1)等于零');
end
K(1)=b1;
A=b/b1;
for m=M:-1:2
    K(m)=A(m);
    J=fliplr(A);
    A=(A-K(m)*J)/(1-K(m)*K(m));
    A=A(1:m-1);
end
```

格型滤波器具体由下面的 latcfilt 函数完成。

```
function [y]=latcfilt(K,x)
%  滤波器的格型实现
% y 为输出序列;K 为格型滤波器(反射)系数数组;x 为输入序列
Nx=length(x)-1;
x=K(1)*x;
M=length(K)-1;
K=K(2:M+1);
fg=[x;[0 x(1:Nx)]];
for m=1:M
    fg=[1,K(m);K(m),1]*fg;
    fg(2,:)=[0 fg(2,1:Nx)];
end
y=fg(1,:);
```

下面给出的 MATLAB 函数 latc2dir 可实现，注意乘积通过两个相应数组的卷积得到，而多项式 J 由对多项式 A 进行 fliplr 运算得到。

```
function [b]=latc2dir(K)
%  全零格型到 FIR 的直接形式的型式转换
```

```
% b 为直接形式的多项式系数(脉冲响应);K 为格型滤波器系数(反射系数)
M=length(K);
J=1;
A=1;
for m=2:1:M
    A=[A,0]+conv([0,K(m)],J);
    J=fliplr(A);
end
b=A*K(1);
```

【例 8-3】一个 FIR 系统的零点分别在 $0.9e^{\pm j\frac{\pi}{3}}$ 和 0.8 处，求其 Lattic 结构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
b=[1,-1.7,1.53,-0.648];
k=tf2latc(b)
b1=latc2tf(k)
```

运行程序，输出如下：

```
k =
   -0.7026
    0.7385
   -0.6480
b1 =
    1.0000   -1.7000    1.5300   -0.6480
```

运行结果表明了 FIR 系统 Lattice 结构与直接型结构之间相互转换的正确性。

8.2.2 全极点格型滤波器

IIR 滤波器的格型结构受限于全极点的系统函数，可根据 FIR 格型结构来开发它。设一个全极点系统函数由下式给定：

$$H(z)=\frac{1}{1+\sum_{m=1}^{M}a_N(m)z^{-m}} \tag{8-10}$$

由此可知

$$H(z)=\frac{1}{A_N(z)} \tag{8-11}$$

显然，它是 FIR 滤波器格型结构的逆系统。所示的 FIR 格型结构的逆系统（除因子 b_0 外），每一级的输入/输出通过顺序递归方程建立联系：

$$\begin{aligned}
&f_N(n)=x(n)\\
&f_{m-1}(n)=f_m(n)-K_m g_{m-1}(n-1),\quad m=N,N-1,\cdots,1\\
&g_m(n)=K_m f_{m-1}(n)+g_{m-1}(n-1),\quad m=N,N-1,\cdots,1\\
&y(n)=f_0(n)=g_0(n)
\end{aligned} \tag{8-12}$$

其中：参数 K_m，$m=1,2,\cdots,M-1$，是全极点格型的反射系数，除了 K_0 等于 1 外，其余各系数可通过上面介绍的全零点格型滤波器 K_m 递归求法求得。

既然与 FIR 格型滤波器相同，IIR 滤波器系数同样可由格型滤波器导出。因此可以使用 MATLAB 中的 dir2latc 函数。应当记住忽略数组 K 中的系数 K_0。类似地，假定如果把数组 K 中的第一个元素设为 K_0=1，则可使用 latc2dir 函数把格型 $\{K_m\}$ 系数转换成直接形式 $\{a_x(m)\}$。

【例 8-4】考虑全极点性滤波器：

$$H(z)=\frac{1}{1+\frac{13}{24}z^{-1}+\frac{5}{8}z^{-2}+\frac{1}{3}z^{-3}} \tag{8-13}$$

求出格型结构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a=[1,13/24,5/8,1/3];
k=dir2latc(a)
```

运行程序，输出如下：

```
k =
    1.0000    0.2500    0.5000    0.3333
```

8.2.3　零极点的 Lattice 结构

一般的 IIR 滤波器既包含零点，又包含极点，它可用极点格型滤波器作为基本构造模块，以格型结构实现，假定 IIR 滤波器的系统函数为

$$H(z)=\frac{\sum_{k=0}^{M}b_M z^{-k}}{1-\sum_{k=1}^{N}a_N z^{-k}}=\frac{B_M(z)}{A_N(z)} \tag{8-14}$$

其中，不失一般性，假定 $N\geqslant M$。为构造一个格型类型结构，首先根据式（8-14）的分母，以系数 K_M，$1\leqslant M\leqslant N$ 实现全极点格型，然后，增加一个梯形部分，把输出看作 $\{g_m(n)\}$ 的加权线性组合，其结果为其有格型梯形结构的零点一极点 IIR 滤波器，输出为

$$y(n)=\sum_{m=0}^{M}C_m g_m(n) \tag{8-15}$$

其中：$\{C_m\}$ 成为确定系统函数 $H(z)$ 零点的梯形系数。$\{C_m\}$ 由下式给定：

$$B_M=\sum_{m=0}^{M}C_m J_m(z) \tag{8-16}$$

其中：$J_m(z)$ 是多项式，由上式可得到递归关系：

$$B_m(z)=B_{m-1}(z)+C_m J_m(z),\quad m=0,2,\cdots,M \tag{8-17}$$

或相对应的，由 $B_m(z)$ 和 $A_m(z)$ 定义得到：

$$C_m=b_m\sum_{i=m+1}^{M}C_i a_i(i-m),\quad m=M,M-1,\cdots,0 \tag{8-18}$$

为实现一般有理 IIR 滤波器的格型梯形结构，首先用递归关系从 $A_N(z)$ 得到格型系数 $\{K_m\}$，然后递归地求解得到梯形系数 $\{C_m\}$，实现分子 $B_N(z)$。

8.3 线性预测滤波器

8.3.1 线性预测滤波器模型

所谓预测就是对未来作出的估计和推断，这往往要对研究对象进行模仿或抽象，这一过程称为建模；用建模手段获得对象的一种表示和体现就称为模型。

信号处理通常是解决如何在噪声中提取信号的问题。因此，需要设计一个模型，该模型有最佳线性过滤特性，当信号与噪声同时输入时，输出端能将信号尽可能精确地恢复出来，而噪声得到了最大的抑制。线性预测法是一种信号建模方法，它将信号的抽取值看成是以前若干时刻采样值的线性组合。该方法实际就是信号的自回归模型，又称最大熵估计。线性预测法主要用于滤波器设计，如语音信息编码、系统辨识和频谱分析等领域。

线性预测的基本任务就是根据已知的前 p 次观测值来预测当前时刻信号 $x(n)$ 的估计值 $\hat{x}(n)$ 。

这里，若假设预测值 $\hat{x}(n)$ 只是根据前一次采样数据 $x(n-1)$ 来作出的，即称为一阶预测：$\hat{x}(n)=-ax(n-1)$ 。其中 $-a$ 为一阶预测系数，系数前“-”号只是为了便于公式推导，这样，预测误差可以写为

$$e(n)=x(n)-\hat{x}(n)=x(n)+ax(n-1) \tag{8-19}$$

根据最小均方误差准则，应使预测误差满足

$$E\{e^2(n)\}=E\{[x(n)-ax(n-1)]^2\} \tag{8-20}$$

满足式（8-20）的系数称为最佳预测系数。为此，对 a 求微分，并令

$$\frac{\partial E\{e^2(n)\}}{\partial a}=2E\left[e(n)\frac{\partial e(n)}{\partial a}\right]=2E[e(n)\cdot x(n-1)] \tag{8-21}$$

则 $2E[e(n)\cdot x(n-1)]=0$ 时误差最小，并进一步写为

$$2E[e(n)\cdot x(n-1)]=E[\{x(n)+ax(n-1)\}x(n-1)]=\varphi_{xx}(x)+a\varphi_{xx}(0)=0 \tag{8-22}$$

得到最佳预测系数 $a_0=-\varphi_{xx}(1)/\varphi_{xx}(0)$ 。这样，一阶线性预测滤波器如图 8-6 所示。

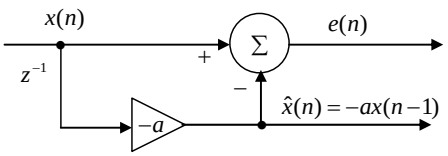


图 8-6　一阶线性预测器

如果线性预测 $\hat{x}(n)$ 还参考了过去 p 个取样值 $x(n-1),x(n-2),\cdots,x(n-p)$ 来预测当前 $x(n)$ ，则构成 p 阶线性预测，表示为

$$\hat{x}(n)=-a_{p1}x(n-1)-a_{p2}x(n-2)-\cdots-a_{pp}x(n-p)=-\sum_{k=1}^{p}a_{pk}x(n-k) \tag{8-23}$$

式（8-23）表明，在预测当前值 $x(n)$ 时，参考了过去 p 个取样值，$-a_{pk}$ 称为第 k 个 p 阶预测系数，系数第一个下标 p 表示 p 阶。信号的预测值 $\hat{x}(n)$ 和真实值 $x(n)$ 值之间存在一个预测误差，可以表示为

$$e_p(n)=x(n)-\hat{x}(n)=-\sum_{k=0}^{p}a_{pk}x(n-k) \tag{8-24}$$

式中：a_{p0} 是常数 1。这样，任务就是根据输入信号序列 $x(n)$ 来产生预测序列 $\hat{x}(n)$ 和预测误差 $e_p(n)$。所以，p 阶线性预测器如图 8-7 所示。所以称式（8-23）为预测滤波器，式（8-24）为预测误差滤波器，这两种滤波器的关系如图 8-7 所示。

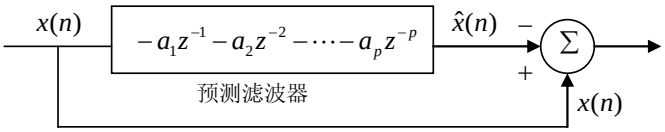


图 8-7　p 阶线性预测器

1．AR 模型

假设系统为 AR 模型，则每个输出 $y(n)$ 是过去的 n_a 个输出的线性组合：

$$y(n)=-\sum_{i=1}^{n_a}a_i y(n-i) \tag{8-25}$$

对于给定的一个系统输出序列，如果采用最小二乘法，就是要找出系统的参数 a_i（i=0,1,⋯,n_a），得到估计值 $\hat{y}(n)=-\sum_{i=1}^{n_a}a_i y(n-i)$，然后使均方误差 $E\{e^2(n)\}=E\{[y(n)-\hat{y}(n)]^2\}$ 取最小值。为此，令

$$\frac{\partial E\{e^2(n)\}}{\partial a_l}=0,\quad l=1,2,\cdots,n_a \tag{8-26}$$

得

$$2E\left\{[y(n)+\sum_{i=1}^{n_a}a_i y(n-i)]y^*(n-l)\right\}=0,\quad l=1,2,\cdots,n_a \tag{8-27}$$

得

$$E\{e(n)\hat{y}^*(n)\}=0 \tag{8-28}$$

于是得

$$\begin{aligned}E\{e^2(n)\}_{\min}&=E\{e(n)[y^*(n)-\hat{y}^*(n)]\}\\&=E\{e(n)y^*(n)\}=E\left\{\left[y(n)+\sum_{i=1}^{n_a}a_i y(n-i)y^*(n)\right]^2\right\}\end{aligned} \tag{8-29}$$

可将式（8-29）和式（8-27）用自相关函数表示，有

$$\begin{cases}R_y(0)+\sum_{i=1}^{n_a}a_iR_y(i)=E\{e^2(n)\}_{\min}\\R_y(l)+\sum_{i=1}^{n_a}a_iR_y(l-i)=0,\quad l=1,2,\cdots,n_a\end{cases} \tag{8-30}$$

写成矩阵形式，得到关于系数 a_i 的方程，就是功率谱估计的 Yule-Walker 方程，形式如下：

$$\begin{bmatrix} R_y(0) & R_y^*(1) & R_y^*(2) & \cdots & R_y^*(n_a) \\ R_y(1) & R_y(0) & R_y^*(1) & \cdots & R_y^*(n_a-1) \\ \vdots & \vdots & \vdots & & \vdots \\ R_y(n_a) & R_y(n_a-1) & R_y(n_a-2) & \cdots & R_y(0) \end{bmatrix} \begin{bmatrix} 1 \\ n_1 \\ \vdots \\ a_{n_a} \end{bmatrix} = \begin{bmatrix} E\{e^2(n)\}_{\min} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8-31}$$

式中：$R_y(n)$ 为信号序列的自相关序列。

以上过程，就是找出系数 a_i 、偏差序列和使实际序列不包含任何相关性，是一个白噪声化的线性预测误差序列。

2．ARMA 模型

AR 模型的解法已经相当成熟，从早期的自相关法到后来的 Burg 方法等，而且许多实际系统大多可用全极点模型来描述，即使有零点，也可以用增加极点数目的方法来接近。但对于有较多的零点的系统，ARMA 模型（零极点模型）仍有广泛应用。

若离散随机过程 $\{x(n)\}$ 服从线性差分方程：

$$x(n)+\sum_{i=1}^{p} a_i x(n-i) = e(n)+\sum_{j=1}^{q} b_j e(n-j) \tag{8-32}$$

式（8-32）中 $e(n)$ 是一离散白噪声，则称 $\{x(n)\}$ 为 ARMA 过程，而式（8-32）差分方程称为 ARMA 模型。系数 $a_1,\cdots,a_p$ 和 $b_1,\cdots,b_q$ 分别称为自回归（AR）参数和滑动平均（MA）参数，p 和 q 分别叫做 AR 阶数和 MA 阶数。

ARMA 模型描述的时不变系统的传递函数定义为

$$H(z)=\frac{B(z)}{A(z)}=\sum_{i=-\infty}^{\infty} h_i z^{-i} \tag{8-33}$$

式中：h_i 为系统的冲激响应系数，可见，系统的极点 $A(z)=0$ 贡献为自回归，而系统零点 $B(z)=0$ 贡献为滑动平均。

对于 ARMA 模型参数的估计算法，早期多采用非线性算法，较为成熟的方法是将 ARMA 问题转化为 AR 模型，求解 AR 模型后，再获得 ARMA 模型参数的估计。

8.3.2 线性预测滤波器设计

1．AR 模型

线性预测方法用于 AR 模型的参数估计。假设一个信号的每个输出样本 $x(k)$ 是过去 n 个样本的线性组合，即

$$x(k)=-a(2)x(k-1)-a(3)x(k-2)-\cdots-a(n+1)x(k-a) \tag{8-34}$$

在 MATLAB 中，可以用函数 lpc 实现对该系统全极点模型系数进行估计，其调用格式如下：

```
[a,g] = lpc(x,p)
```

其中：x 是要建模的信号，n 为模型的阶数；在返回值中，a 为全极点 IIR 滤波器系数，g 为滤波器增益。如果不指定 n，则 lpc 使用默认值 n=length(x)−1。

【例 8-5】试用一个 3 阶前向预测器来估计模型系数，并与最初的信号相比较。

其实现的 MATLAB 程序代码如下：

```
%首先，需要产生一批数据，用白噪声驱动自回归模型，并用 AR 输出最后 4096 点数据
```

```
randn('state',0);
noise=randn(50000,1);          %正态高斯白噪声
x=filter(1,[1 1/2 1/3 1/4],noise);
x=x(45904:50000);
%调用线性预测函数 lpc，计算预测系数，并估算预测误差以及预测误差的自相关
a=lpc(x,3);
est_x=filter([0 -a(2:end)],1,x);  %信号估算
e=x-est_x;                        %预测误差
[acs,lags]=xcorr(e,'coeff');      %预测误差的 ACS
%比较预测信号和原始信号，如图 8-8 所示
plot(1:97,x(4001:4097),1:97,est_x(4001:4097),'--');
title('Original Signal vs.LPC Estimate');
xlabel('Sample Number');ylabel('Amplitude');
grid on;
legend('Original Signal','LPC Estimate');
%分析预测误差的自相关，如图 8-9 所示
plot(lags,acs);
title('Autocorrelation of the Prediction Error');
xlabel('Lags');
ylabel('Normalized Value');
grid on;
```

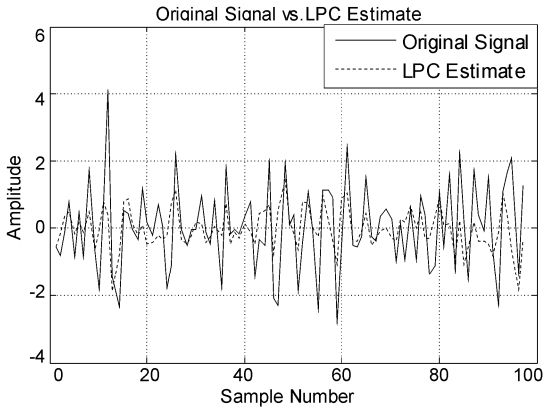


图 8-8　原始信号与预测信号

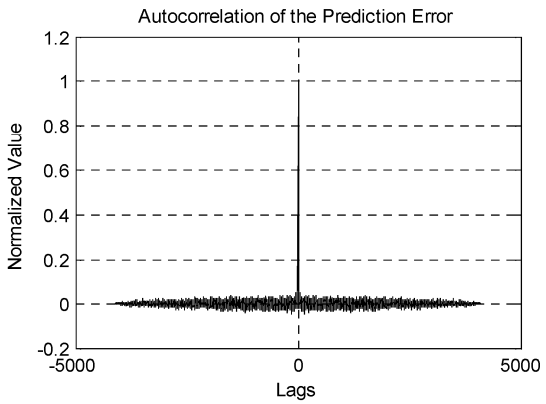


图 8-9　预测错误的自相关

2．ARMA 模型（Prong 法）

Prony 方法用于时域 IIR 滤波器设计，用指定数量的极点和零点为信号序列建模。在 MATLAB 信号处理工具箱中提供了函数 prony，函数的调用格式如下：

[b,a] = prony(h,n,m)

参数 n 和 m 分别表示滤波器的分子阶数和分母阶数，h 是时域的脉冲响应序列。如果 h 的长度小于 n 和 m 的最大值，h 将用零填补。调用后，返回滤波器系数矢量 a 和 b，长度分别为 n+1 和 m+1。滤波器系数在 z 域中按降序排列，公式如下：

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{a(1)+a(2)z^{-1}+\cdots+a(m+1)z^{-m}} \tag{8-35}$$

当然，返回的滤波器不一定稳定，但是如果数据序列是一个给定阶数的自回归滑动平均过

程，则 prong 方法有可能精确地恢复出复数。

【例 8-6】建立一个 4 阶巴特沃思原始滤波器，并从脉冲响应巴特沃思滤波器中恢复出它的系数。

其实现的 MATLAB 程序代码如下：

```
>> [b,a] = butter(4,0.2)
b =
    0.0048    0.0193    0.0289    0.0193    0.0048
a =
    1.0000   -2.3695    2.3140   -1.0547    0.1874
>> h = filter(b,a,[1 zeros(1,25)]);
>> [bb,aa] = prony(h,4,4)
bb =
    0.0048    0.0193    0.0289    0.0193    0.0048
aa =
    1.0000   -2.3695    2.3140   -1.0547    0.1874
```

以上结果表明，用函数 prong 所建立的新滤波器脉冲响应和原始滤波器 h 完全吻合。

3．ARMA 模型（Steiglitz-McBride 法）

对于系统函数如式（8-36）所列的滤波器，可以采用 Steiglitz-McBride 快速迭代算法对滤波器的参数进行估计。

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(nb+1)z^{-nb}}{a(1)+a(2)z^{-1}+\cdots+a(ma+1)z^{-ma}} \tag{8-36}$$

利用 Steiglitz-McBride 迭代法进行建模，将使系统 $H(z)$ 的脉冲响应 x' 和输入信号 x 的均方误差最小。

在 MATLAB 信号处理工具箱中，可以使用函数 stmcb 来计算一个线性模型，其调用格式如下：

[b,a] = stmcb(h,nb,na)

[b,a] = stmcb(y,x,nb,na)

[b,a] = stmcb(h,nb,na,niter)

[b,a] = stmcb(y,x,nb,na,niter)

[b,a] = stmcb(h,nb,na,niter,ai)

[b,a] = stmcb(y,x,nb,na,niter,ai)

【例 8-7】建立一个 6 阶的巴特沃思脉冲响应滤波器，并利用 stmcb 函数对该滤波器进行预测。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[b,a] = butter(6,0.2);              %建立一个 6 阶巴特沃思脉冲响应
h = filter(b,a,[1 zeros(1,100)]);
freqz(b,a,128)                      %幅频—相频特性
% 利用 stmcb 函数进行线性预测
>> [bb,aa] = stmcb(h,4,4);
freqz(bb,aa,128)
```

运行程序，效果如图 8-10 及图 8-11 所示。

此外，也可以利用 stmcb 函数，对给定输出和输入序列的系数进行估计识别。比如存在某

系统，其输入序列为 x=[1 1 1 1]，通过系统滤波器后，输出为 y，即

```
>> x=[1 1 1 1];
>> y=filter(1,[1,1],x);            %产生输出信号
```

利用 stmcb 函数对该系统进行识别，有

```
>> [b,a]=stmcb(y,x,0,1)
```

运行程序，输出如下：

```
b =
    1.0000
a =
    1.0000    1.0000
```

可见，stmcb 正确识别了由 x 输出 y 的系统，得到的参数与 filter 中的参数相同。

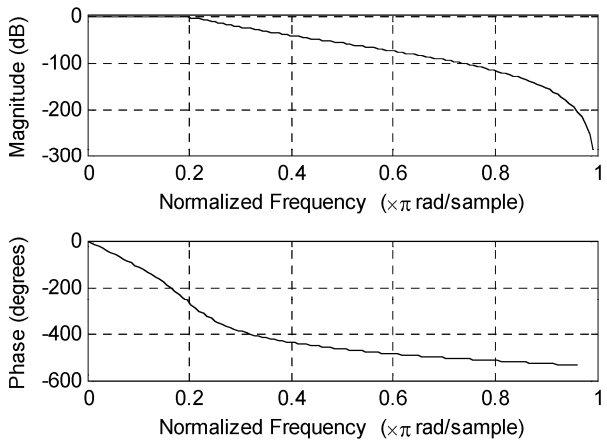


图 8-10　6 阶巴特沃思滤波器

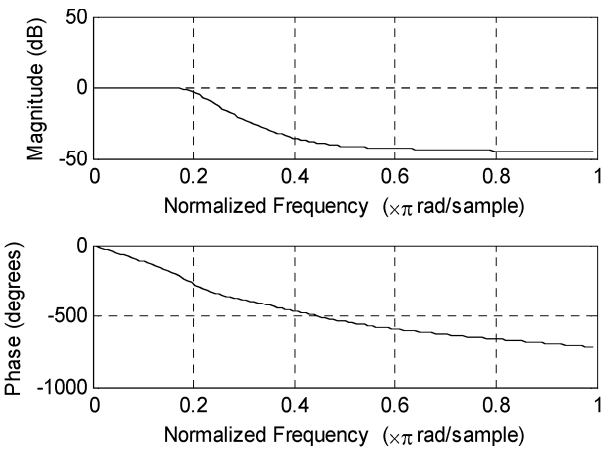


图 8-11　stmcb 预测的滤波器特性

第 9 章　随机信号及参数建模

9.1　随机信号基本处理

9.1.1　随机信号的基本定义

一维离散随机信号就是把时间 n 作为参数的随机变量的集合依时序排列得到的序列，又称为随机过程，若将 $X(n)$ 记作随机变量，则 $\{X(n)\}$ 表示一个随机信号序列或离散随机过程。

9.1.2　离散随机过程的时域统计描述

1．概率分布

对于一个随机过程 $\{X(n)\}$，可以用概率分布函数来描述，即事件 $\{X(n)<x\}$ 的概率为

$$P\{X(n)<x\}=F(x,n) \tag{9-1}$$

$F(x,n)$ 并没有完整地描述由随机变量 $X(n)$ 构成的信号序列 $\{X(n)\}$，因为不同时刻的 $X(n)$ 相互之间也有关系，未必是独立的。要完整地了解 $\{X(n)\}$ 的统计特性，必须对任何 k 个时刻 $n_1,n_2,\cdots,n_k$，知道 $X(n_1),X(n_2),\cdots,X(n_k)$ 的联合分布函数：

$$F(x_1,x_2,\cdots,x_k;n_1,n_2,\cdots,n_k)=P\{X(n_1)<x_1,X(n_2)<x_2,\cdots,X(n_k)<x_k\} \tag{9-2}$$

这就给出了随机信号序列 $\{X(n)\}$ 的最一般的概率模型。

在滤波理论中，要处理的一个基本情况是平稳随机信号序列，“平稳”的含义是 $\{X(n)\}$ 的统计特性不随时间的推移而变化，也就是一种概率意义下的移位不变性。具体地讲，就是分布函数 $F(x,n)$ 不随时间 n 而变化，应该有

$$F(x,n)=F(x) \tag{9-3}$$

而对于联合分布函数，当 k 个时刻 $n_1,n_2,\cdots,n_k$ 都推移 i 个时间单位，成为 $n_1+i,n_2+i,\cdots,n_k+i$ 时，相应的 k 个随机变量 $X(n_1+i),X(n_2+i),\cdots,X(n_k+i)$ 仍应有相同的联合分布函数，即

$$F(x_1,x_2,\cdots,x_k;n_1+i,n_2+i,\cdots,n_k+i)=F(x_1,x_2,\cdots,x_k;n_1,n_2,\cdots,n_k) \tag{9-4}$$

以上两式给出的平稳性要求在概率论意义上是严格而完备的，又称“强平稳性或严格平稳性”。由于在许多情况下对一个随机变量的了解只要知道它的数学期望和方差就够了，不一定要完整地掌握分布函数；对于多个随机变量，另外再知道任何两个随机变量之间的协方差就够了，这样可以给出一个较为宽松的要求，即

$$E\{X(n)\}=a\ （常数） \tag{9-5}$$

对于任何 n 成立，以及

$$R_x(n+i,m+i)=R_x(n,m) \tag{9-6}$$

其中

$$R_x(n,m)=E\{X(n)X^*(m)\} \tag{9-7}$$

对任何 n，m，j 成立。这里 E 表示取数学期望，R_x 表示 x 的自相关函数。上式还可以化简，令

m=0，则有

$$R_x(n+i,i)=R_x(n,0)=R'_x(n) \tag{9-8}$$

式（9-8）的意义是：两个相距 n 个时间单位的随机信号变量之间的协方差与两个信号变量在时间轴上的绝对位置无关。满足上述数学期望和自相关函数给出的平稳性要求的随机信号序列称为“弱平稳随机信号序列”或“广义平稳随机信号序列”。

除了随机信号序列的平稳性，另一个重要的性质是信号序列的“各态历经性”。粗略地说，若一个过程的完全统计特性由总体中的任意样本函数来获得，则此过程称为遍历性过程或各态历经过程。具体而言，对于平稳随机信号 $X(n)$ 来说，如果它的所有样本函数在某一固定时刻的一阶和二阶特性和单一样本函数在长时间内的统计特性一致，则称 $X(n)$ 为各态遍历信号，且其均值与自相关函数都能够使用单一的样本函数作时间平均来计算。

在实际工作中，往往假定信号是平稳的，而且是各态遍历的，在此不加说明地认为所讨论的信号都是平稳的和各态遍历的。

2．概率分布特性的特征量

1）均值

均值也就是随机变量 $X(n)$ 的数学期望，在 MATLAB 信号处理工具箱中提供 mean(x)函数实现均值，它的计算公式为

$$\overline{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$$

2）均方值

均方值是随机变量 $X(n)$ 平方的数学期望，其计算公式为

$$E\{x^2\}=\frac{1}{n}\sum_{i=1}^{n}x_i^2$$

3）方差

方差的定义是：$E\{(x-Ex)^2\}=\dfrac{1}{n-1}\sum_{i=1}^{n}(x_i-m_x)^2$，方差平方根称为标准差或均方根。

4）自相关序列与互相关序列

一个平稳随机序列的自相关序列定义为

$$R_x(m)=E\{X(n)X^*(n+m)\}$$

其互相关序列定义为

$$R_{xy}(m)=E\{X(n)Y^*(n+m)\}$$

5）自协方差序列与互协方差序列

一个平稳随机序列的自协方差序列定义为

$$C_x(m)=E\{[X(n)-\mu_x][X(n+m)-\mu_x]^*\}$$

互协方差序列定义为

$$C_{xy}(m)=E\{[X(n)-\mu_x][Y(n+m)-\mu_y]^*\}$$

协方差序列与相关序列关系为

$$C_{xy}(m)=R_{xy}(m)-\mu_x\mu_y^*$$

【例 9-1】计算长度 N=100000 的正态分布高斯随机信号的均值、均方值、均方根值、方差和均方差。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=100000;          %数据个数
randn('state',0);  %设置产生随机数的状态
y=randn(1,N);      %产生一个随机序列
disp('平均值:');
yMean=mean(y)      %计算随机序列的均值
disp('平方值:');
y2p=y*y'/N         %计算其均方值,这里利用了矩阵相乘的算法
disp('平方根:')
ysq=sqrt(y2p)      %计算其均方根值
disp('标准差:')
ystd=std(y,1)      %相当于 ystd=sqrt(sum((y-yMean).^2)/(N-1))
disp('方差:');
yd=ystd.*ystd
```

运行程序，输出如下：

```
平均值:
yMean =       0.0032
平方值:
y2p =       1.0073
平方根:
ysq =       1.0037
标准差:
ystd =       1.0037
方差:
yd =       1.0073
```

注意，函数 s=std(x, flag)计算标准差。x 为向量或矩阵；s 为标准差；flag 为控制符，用来控制标准算法。当 flag=0（或默认）时，按下式计算无偏标准差：

$$S=\left[\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)^2\right]^{\frac{1}{2}}$$

当 flag=1 时，按下式计算有偏标准差：

$$S=\left[\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_x)^2\right]^{\frac{1}{2}}$$

【例 9-2】求带白噪声干扰的频率为 10Hz 的正弦信号和白噪声信号的自相关函数并进行比较。其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=1000; Fs=500;                                  %数据长度和采样频率
n=0:N-1; t=n/Fs;                                 %时间序列
Lag=100;                                         %延迟样点数
randn('state',0);                                %设置产生随机数的初始状态
x=sin(2*pi*10*t)+0.6*randn(1,length(t));         %原始信号
[c,lags]=xcorr(x,Lag,'unbiased');                %对原始信号进行无偏自相关估计
subplot(2,2,1); plot(t,x);                       %绘制原始信号 x
xlabel('时间/s'); ylabel('x(t)');
```

```
title('带噪声周期信号');
grid on;
subplot(2,2,2);plot(lags/Fs,c);          %绘制 x 信号自相关,lags/Fs 为时间序列
xlabel('时间/s'); ylabel('Rx(t)');
title('带噪声周期信号的自相关');
grid on;
%  信号 x1
x1=randn(1,length(x));                   %产生一与 x 长度一致的随机信号
[c,lags]=xcorr(x1,Lag,'unbiased');       %求随机信号 x1 的无偏自相关
subplot(2,2,3); plot(t,x1);              %绘制随机信号 x1
xlabel('时间/s'); ylabel('x1(t)');
title('噪声信号');
grid on;
subplot(2,2,4); plot(lags/Fs,c);         %绘制随机信号 x1 的无偏自相关
xlabel('时间/s'); ylabel('Rx1(t)');
title('噪声信号的自相关');
grid on
```

运行程序，效果如图 9-1 所示。

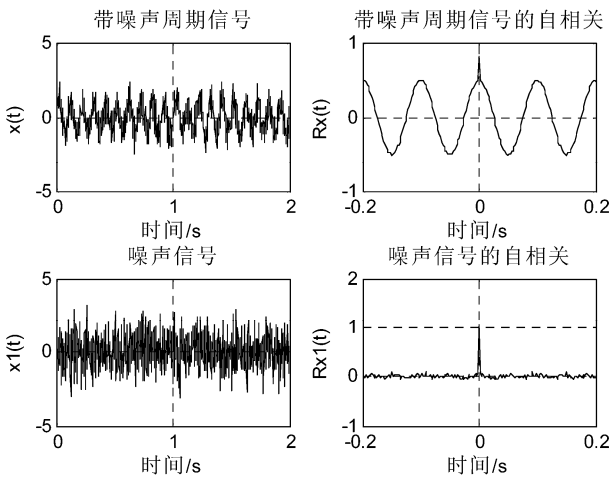


图 9-1　带有白噪声的周期信号和噪声信号的自相关

【例 9-3】比较一下某函数的互相关函数和互协方差函数。

其实现的 MATLAB 程序代码如下：

```
clear all;
t=1:20;x=t.^2;
y=(t+2).^2;
R=xcorr(x,y);
c=xcov(x,y);
n=1:length(c);
subplot(211);stem(n,c);
title('互协方差');
subplot(212);stem(n,R);
title('互相关');
```

运行程序效果如图 9-2 所示。

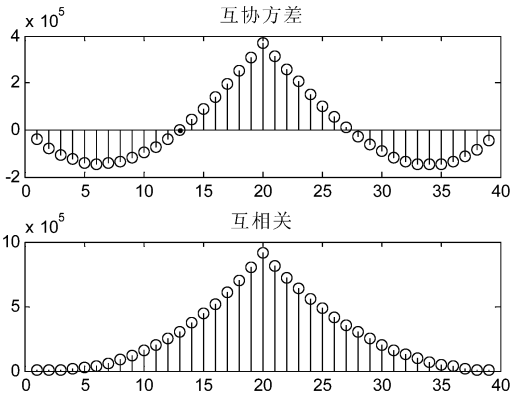


图 9-2　两平方函数的互相关与互协方差函数

9.1.3　离散随机过程的频域统计描述

无限能量信号本身的 Z 变换与傅里叶变换是不存在的，但离散随机序列的自协方差序列与自相关序列存在如下性质，即

$$\begin{cases}\lim\limits_{m\to\infty} C_x(m)=0\\ \lim\limits_{m\to\infty} R_x(m)=m_x^2\end{cases} \tag{9-9}$$

当 $m_x=0$ 时，有 $R_x(m)\to 0$。因此，离散随机序列的自协方差序列和自相关序列（$m_x=0$）的 Z 变换和傅里叶变换是存在的。

设 $C_x(m)$ 的 Z 变换为 $\Gamma_x(z)$，则按 Z 变换及反变换的定义有

$$\begin{cases}\Gamma_x(z)=\sum\limits_{m=-\infty}^{\infty} C_x(m)z^{-m}, \quad R_{z^-}<|z|<R_{z^+}\\ C_x(m)=\dfrac{1}{2\pi \mathrm{j}}\oint_c \Gamma_x(z)z^{m-1}\mathrm{d}z\end{cases} \tag{9-10}$$

式中：c 是 $\Gamma_x(z)$ 收敛域内的一条闭合围线，且由自协方差或自相关的性质可知，收敛域一定包含单位圆。

当 $m_x=0$ 时，则有

$$E\{X^2(n)\}=C_x(0)=R_x(0)=\frac{1}{2\pi}\int_{-\pi}^{\pi}P_x(\omega)\mathrm{d}\omega \tag{9-11}$$

$E\{X^2(n)\}$ 表示信号的平均功率，于是式（9-11）说明了 $P_x(\omega)$ 在 $[-\pi,\pi]$ 频域区间内的积分面积正比于信号 $\{X(n)\}$ 的平方功率。因此，$P_x(\omega)$ 被称为 $\{X(n)\}$ 的功率谱密度。

当 $m_x=0$ 时，$R_x(m)=C_x(m)$，故此时的自相关函数 $R_x(m)$ 与功率谱密度 $P_x(\omega)$ 是一对傅里叶变换对，即

$$\begin{cases}P_x(\omega)=\sum\limits_{m=-\infty}^{\infty} R_x(m)\mathrm{e}^{-\mathrm{j}\omega m}\\ P_x(m)=\dfrac{1}{2\pi}\int_{-\pi}^{\pi}P_x(\omega)\mathrm{e}^{\mathrm{j}\omega m}\mathrm{d}\omega\end{cases} \tag{9-12}$$

如果 $m_x \neq 0$，当 $m \to \infty$ 时，$R_x(m) \to m_x^2$，这时自相关序列的傅里叶变换不存在，除非把傅里叶变换定义推广，允许功率谱在 $\omega = 0$ 处有一冲激。

功率谱密度 $P_x(\omega)$ 的性质表明，它是偶函数且函数值必定是非负实数。

类似地，两个离散随机序列 $\{X(n)\}$ 和 $\{Y(n)\}$ 的互功率谱密度定义为

$$P_{xy}(\omega) = \Gamma_{xy}(\mathrm{e}^{\mathrm{j}\omega})$$

9.2　功率谱估计

我们已经知道一个随机信号本身的傅里叶变换是不存在的，因此无法像确定性信号那样用数学表达式来精确地描述它，而只能用各种统计平均量来表征它。其中，自相关函数最能完整地表征它的特定统计平均量值。而一个随机信号的功率谱密度正是自相关函数的傅里叶变换，可以用功率谱密度来表征它的统计平均谱特性。所以，要在统计意义下描述一个随机信号，就需要估计它的功率谱密度（Power Spectral Density，PSD）。功率谱估计在其他应用中也有十分重要的作用，如设计最优线性滤波器、测量噪声频谱、检测埋没在宽带噪声中的窄带信号，以及用噪声激励法估计线性系统的参数等。因此，随机信号的谱估计是当前信号处理中一个重要的研究方向。

功能谱估计有多种方法，一般可以分为参数化方法和非参数化方法。非参数化方法中较为常用的是 Welch 方法，它属于经典谱估计的一类——周期图法。此外，MATLAB 信号处理工具箱还提供了其他一些现代的非参数化方法，包括 Multiaper 方法、MUSIC（Multiple Signal Classification）或特征向量方法，这些方法非常适合线谱的估计。相比非参数化方法，参数化方法则主要围绕 ARMA（自回归滑动平均）模型的参数估计问题来计算信号的功率谱，属于现代谱估计方法，其频率分辨性能要优于经典谱估计，MATLAB 提供的主要方法有 Yule-Walker 自回归方法，基于线性预测误差最小的 Burg 方法，以及协方差方法与改进的协方差方法。

9.2.1　经典功率谱估计法

平稳随机信号的功率谱密度（PSD）是相关序列的离散傅里叶变换，即

$$P_{xx}(\omega) = \sum_{m=-\infty}^{\infty} r_{xx}(m)\mathrm{e}^{-\mathrm{j}\omega m} \tag{9-13}$$

它实际上是互功率谱密度（CSD）函数的特殊情况，互谱密度定义为

$$P_{xy}(\omega) = \sum_{m=-\infty}^{\infty} r_{xy}(m)\mathrm{e}^{-\mathrm{j}\omega m} \tag{9-14}$$

功率谱密度的单位一般为 dBw/Hz 或 dBm/Hz。

谱估计方法分为两大类：非参数化方法和参数化方法。经典谱估计法属于非参数化方法，它又可以分为用随机序列求谱的自相关法和将序列直接用 FFT 求谱的直接法。经典谱估计法对所得到的数据序列只进行线性运算，因而又称为线性谱分析法。

1．直接法

直接法即周期图法，于 1989 年由舒斯特提出，直接由傅里叶变换得到：将随机信号 $x(n)$ 的 N 点样本值 $x_N(n)$ 视为能量有限信号，取其傅里叶变换，得到 $x_N(\mathrm{e}^{\mathrm{j}\omega})$；再取其幅值的平方，并除以 N 作为 $x(n)$ 的真实功率谱 $P(\mathrm{e}^{\mathrm{j}\omega})$ 的估计，即

$$\hat{P}(e^{j\omega}) = \frac{1}{N}\left|X_N(\omega)\right|^2 \tag{9-15}$$

在 MATLAB 信号处理工具箱中，提供了 periodogram 函数实现直接功率谱估计，其调用格式如下：

```
[Pxx,w] = periodogram(x)
[Pxx,w] = periodogram(x,window)
[Pxx,w] = periodogram(x,window,nfft)
[Pxx,w] = periodogram(x,window,w)
[Pxx,f] = periodogram(x,window,nfft,fs)
[Pxx,f] = periodogram(x,window,f,fs)
[Pxx,f] = periodogram(x,window,nfft,fs,'range')
[Pxx,w] = periodogram(x,window,nfft,'range')
periodogram(...)
```

【例 9-4】采用 periodogram 函数来计算功率谱。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',0);
Fs = 1000;
t = 0:1/Fs:.3;
x = cos(2*pi*t*200)+0.1*randn(size(t));
periodogram(x,[],'twosided',512,Fs);
xlabel('频率/kHz');
ylabel('相对功率谱密度(dB/Hz)');
title('直接法');
```

运行程序，效果如图 9-3 所示。

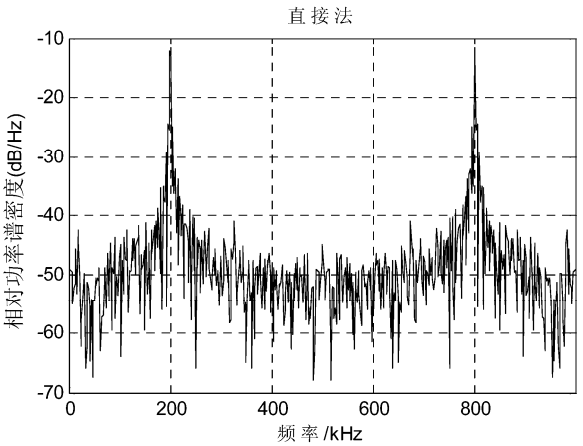


图 9-3　用 periodogram 函数计算功率谱效果

【例 9-5】用傅里叶变换算法求信号 $x = \sin(2\pi f_1 t) + 2\sin(2\pi f_2 t) + \omega(t)$ 的功率谱。其中，$f_1 = 50\text{Hz}$，$f_2 = 120\text{Hz}$，$\omega(t)$ 为白噪声，采样频率 $F_s = 1000\text{Hz}$。①信号长度 $N = 256$；②信号长度 $N = 1024$。

其实现的 MATLAB 程序代码如下：

```
clear all;
Fs=1000;
%  第一种情况:N=256
N=256;Nfft=256;   %数据长度和 FFT 所用的数据长度
n=0:N-1;t=n/Fs;     %采用的时间序列
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);%带有白噪声的信号
Pxx=10*log10(abs(fft(xn,Nfft).^2)/N);
% Fourier 振幅谱平方的平均值,并转换为 dB
f=(0:length(Pxx)-1)*Fs/length(Pxx);   %给出频率序列
subplot(211);plot(f,Pxx);                  %绘制功率谱曲线
xlabel('频率/Hz');ylabel('功率谱/dB');
title('周期图 N=256');
grid on;
%  第二种情况:N=1024
Fs=1000;
N=1024;Nfft=1024;   %数据长度和 FFT 所用的数据长度
n=0:N-1;t=n/Fs;         %采用的时间序列
%带有白噪声的信号
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);
Pxx=10*log10(abs(fft(xn,Nfft).^2)/N);
% Fourier 振幅谱平方的平均值,并转换为 dB
f=(0:length(Pxx)-1)*Fs/length(Pxx);   %给出频率序列
subplot(212);plot(f,Pxx);                  %绘制功率谱曲线
xlabel('频率/Hz');ylabel('功率谱/dB');
title('周期图 N=1024');
grid on;
```

运行程序，效果如图 9-4 所示。

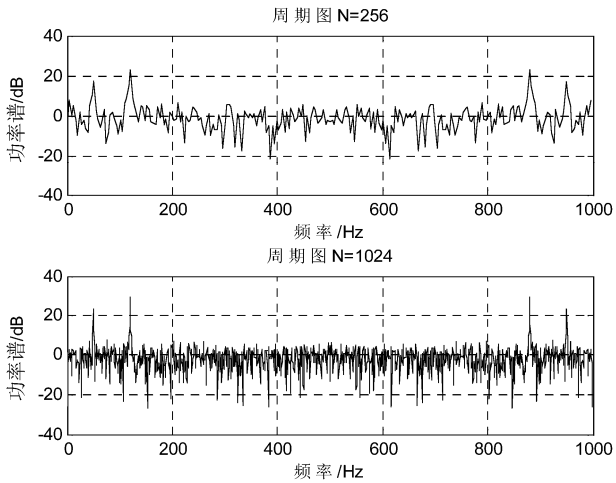


图 9-4　含有噪声的信号的功率谱

2．间接法

间接法，又称为自相关法或 BT 法，在 1958 年由布莱克曼与图基首先开拓。间接法的理论基础是维纳—辛钦定理，它是由随机信号的 N 个观察值 $x(0),x(1),\cdots,x(N-1)$，估计出自相关函数 $R_N(m)$，然后再求 $R_N(m)$ 的傅里叶变换作为功率谱的估计。

$$\hat{S}(e^{j\omega})=\sum_{m=-M}^{N}R_N(m)e^{-j\omega m}\quad |M|\leqslant N-1 \tag{9-16}$$

【例 9-6】利用间接法重新计算例 9-4 中噪声信号的功率谱。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',0);
Fs = 1000;
NFFT=1024;
n=0:1/Fs:1;
t = 0:1/Fs:.3;
x = cos(2*pi*t*200)+0.1*randn(size(t));
Cx=xcorr(x,'unbiased');       %计算序列的自相关函数
Cxk=fft(Cx,NFFT);             %求解 PSD
pxx=abs(Cxk);
t=0:round(NFFT/2-1);
k=t*Fs/NFFT;                  %横坐标为频率,单位为 Hz
P=10*log(pxx(t+1));           %纵坐标为相对功率谱密度,单位为 dB/Hz
plot(k,P);
xlabel('频率/kHz');
ylabel('相对功率谱密度(dB/Hz)');
title('间直接法');
grid on;
```

运行程序，效果如图 9-5 所示。

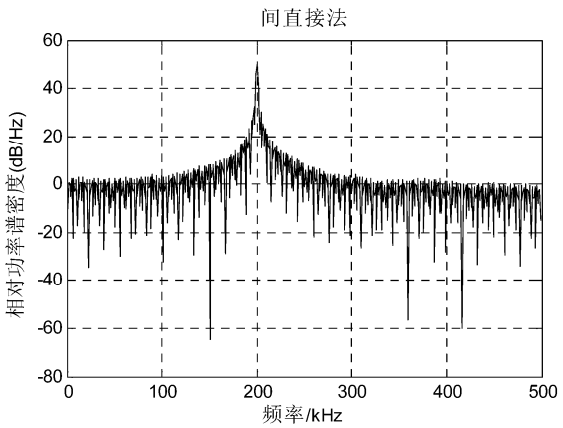


图 9-5　利用间接法重新计算例 9-4 中噪声信号的功率谱

3．基于经典谱估计的系统辨识

非参数化的系统辨识，是经典谱估计的一个很有用的方法。假定需要辨识的线性系统为 H，输入信号为 $x(n)$，输出信号为 $y(n)$，则计算输入信号和输出信号的互相关系数，可得

$$R_{xy}(m)=E\{X(n)Y(n+m)\}=R_x(m)*H(m) \tag{9-17}$$

将等式两边进行傅里叶变换，得

$$P_{xy}(\omega)=P_x(\omega)H(\omega) \tag{9-18}$$

估计出互谱密度 $P_{xy}(\omega)$ 和功率谱密度 $P_x(\omega)$，就可以得到 $H(\omega)$ 的辨识结果，即

$$\hat{H}(\omega)=\frac{\hat{P}_{xy}(\omega)}{\hat{P}_{x}(\omega)} \tag{9-19}$$

在 MATLAB 信号处理工具箱中，提供了 tfe 函数实现基于经典谱估计的系统辨识，其调用格式如下：

Txy = tfe (x,y)

Txy = tfe (x,y,window)

Txy = tfe (x,y,window,noverlap)

[Txy,W] = tfe (x,y,window,noverlap,nfft)

[Txy,F] = tfee(x,y,window,noverlap,nfft,fs)

[…] = tfe (x,y,…,'whole')

tfe (…)

【例 9-7】采用 tfe 函数进行系统的辨识，并与理想结果进行比较。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=2000;
NFFT=256;
n=0:1/Fs:1;
x=randn(size(n));
b=ones(1,5)/5;
y=filter(b,1,x);
[h,f]=tfe(x,y,NFFT,Fs,256,128,'none');
h0=freqz(b,1,f,Fs);
subplot(2,1,1);plot(f,abs(h0));
grid on;
title('理想传递函数的幅度');
xlabel('频率/Hz');
subplot(2,1,2);plot(f,abs(h));
title('估计的传递函数幅度');
xlabel('频率/Hz')
grid on;
```

运行程序，效果如图 9-6 所示。

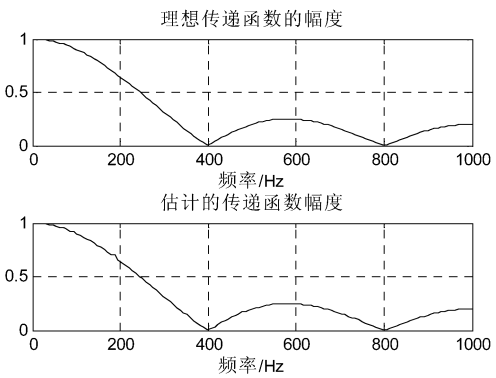


图 9-6　tfe 函数进行系统的辨识与理想结果进行比较

9.2.2 改进的直接法估计

直接法和间接法的方差性能很差，而且当数据长度太大时，谱曲线起伏加剧；若数据长度太小，则谱的分辨率又不好，所以需要改进。改进的直接谱估计方法主要有 Bartlett 法和 Welch 法。

1．Bartlett 法

Bartlett 法通过多个重叠数据段周期图的平均来改进周期方法的性能。把数据序列 $x(n)(0\leqslant n\leqslant N-1)$ 分为 K 段，每段有 M 个样本，$N=KM$，形成式（9-20）所表示的序列段：

$$x^{(i)}(n)=x(n+iM-N),\quad 0\leqslant n\leqslant M-1,1\leqslant i\leqslant K \tag{9-20}$$

计算如下 K 个周期图，得

$$I_M^{(i)}(\omega)=\frac{1}{M}\left|\sum_{n=0}^{M-1}x^{(i)}(n)\mathrm{e}^{-\mathrm{j}\omega}\right|,\quad 1\leqslant i\leqslant K \tag{9-21}$$

如果 $m\geqslant M$ 时，$R_x(m)$ 很小，那么就可以假设诸周期图 $I_M^{(i)}(\omega)$ 彼此独立，于是由这 K 个单独的周期图，可以定义一个平均周期图 $B_x(\omega)$，即

$$B_x(\omega)=\frac{1}{K}\sum_{i=1}^{K}I_M^{(i)}(\omega) \tag{9-22}$$

这种求谱估计的方法，就是 Bartlett 谱估计方法。用 FFT 实现这种估计最方便，可以调用任何点数来计算 $x^{(i)}(n)$ 的周期图，当点数大于数据段长度 M 时，可以填充零值补满。虽然周期图取平均值的方法的方差比较小，但是它仍然是一种有偏渐近一致估计。

在 MATLAB 工具箱中，函数 psd 用来实现 Bartlett 平均周期图方法的功率谱估计，而函数 csd 用来实现信号间的互功率谱估计。

1）psd 函数

功能：实现 Bartlett 平均周期图方法的功率谱估计，其调用格式如下：

```
pxx=psd(x, NFFT, Fs, window)
[pxx, f]=psd(x, NFFT, Fs, window, noverlap)
[pxx, pxxc, f]=psd(x, NFFT, Fs, window, noverlap, p)
psd(x, …, dflag)
psd(…)
```

【例 9-8】在置信度为 95%的区间上估计有色噪声 x 的 PSD。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=1000;
NFFT=256;
p=0.95;                    %置信区间
[b,a]=ellip(6,2,50,0.2);   %设计 6 阶椭圆型滤波器
r=randn(4096,1);
x=filter(b,a,r);           %对白噪声滤波得到信号 x
psd(x,NFFT,Fs,[],0,p);     %PSD 估计
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
```

运行程序，效果如图 9-7 所示。

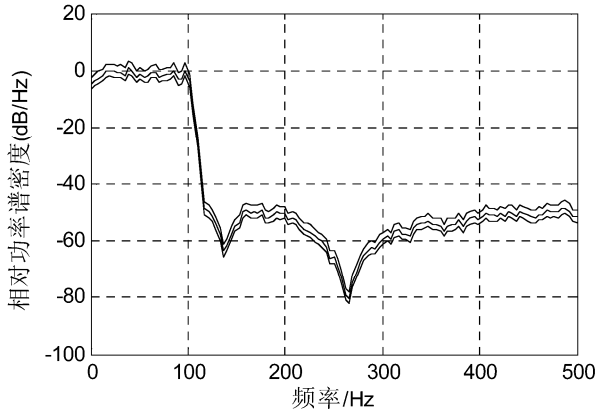


图 9-7 有色噪声 x 的 PSD

2）csd 函数

功能：信号间的互功率谱估计，其调用格式如下：

pxy=csd(x, y, NFFT, Fs, window)

[pxy, f]=csd(x, y, NFFT, Fs, window, noverlap)

[pxy, pxyc, f]=csd(x, y, NFFT, Fs, window, noverlap)

csd(x, y,…, dflag)

csd(…)

【例 9-9】在置信度为 95%的区间上估计两个有色噪声 x，y 之间的 CSD。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=1000;
NFFT=1024;
p=0.95;        %置信区间
H=fir1(40,0.5,boxcar(41));
h=ones(1,10)/sqrt(10);
r=randn(4096,1);
x=filter(h,1,r);
y=filter(H,1,x);
csd(x,y,NFFT,Fs,triang(500),0,p);
xlabel('频率/Hz');ylabel('相对互功率谱密度(dB/Hz)');
```

运行程序，效果如图 9-8 所示。

【例 9-10】对例 9-5 中的信号序列，用两种平均周期图法求信号的功率谱密度估计。

其实现的 MATLAB 程序代码如下：

```
clear all;
Fs=1000;                %数据的采样点
% 运用信号不重叠分段估计功率谱
N=1024;N1=256;     %数据点数
n=0:N-1;t=n/Fs;       %分段间隔,时间序列
randn('state',0);        %设置产生随机数的初始状态
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);
```

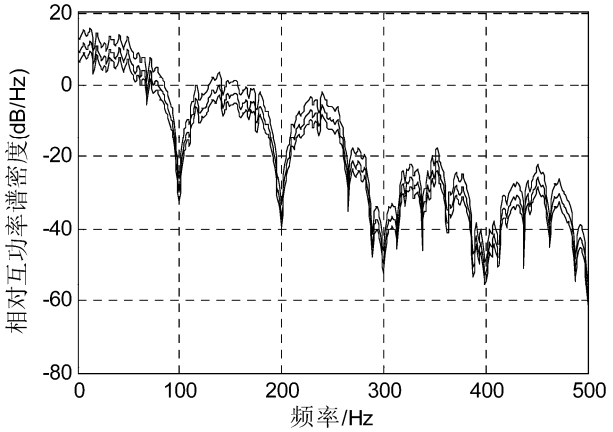


图 9-8　两个有色噪声 x, y 之间的 CSD

```
% 带噪声的原始信号
Pxx1=abs(fft(xn(1:256),N1).^2)/N1;          %第一段功率谱
Pxx2=abs(fft(xn(257:512),N1).^2)/N1;        %第二段功率谱
Pxx3=abs(fft(xn(513:768),N1).^2)/N1;        %第三段功率谱
Pxx4=abs(fft(xn(769:1024),N1).^2)/N1;       %第四段功率谱
Pxx=10*log10((Pxx1+Pxx2+Pxx3+Pxx4)/4);      %平均得到整个序列功率谱
f=(0:length(Pxx)-1)*Fs/length(Pxx);         %给出功率谱对应的频率
subplot(211);plot(f(1:N1/2),Pxx(1:N1/2));   %绘制功率谱曲线
xlabel('频率/Hz');ylabel('功率谱/dB');
title('平均周期图(无重叠)N=4*256');
grid on;
%运用信号重叠分段估计功率谱
Pxx1=abs(fft(xn(1:256),N1).^2)/N1;          %第一段功率谱
Pxx2=abs(fft(xn(129:384),N1).^2)/N1;        %第二段功率谱
Pxx3=abs(fft(xn(257:512),N1).^2)/N1;        %第三段功率谱
Pxx4=abs(fft(xn(385:640),N1).^2)/N1;        %第四段功率谱
Pxx5=abs(fft(xn(513:768),N1).^2)/N1;        %第五段功率谱
Pxx6=abs(fft(xn(641:896),N1).^2)/N1;        %第六段功率谱
Pxx7=abs(fft(xn(769:1024),N1).^2)/N1;       %第七段功率谱
%平均得到整个序列功率谱
Pxx=10*log10((Pxx1+Pxx2+Pxx3+Pxx4+Pxx5+Pxx6+Pxx7)/7);
f=(0:length(Pxx)-1)*Fs/length(Pxx);         %给出功率谱对应的频率
subplot(212);plot(f(1:N1/2),Pxx(1:N1/2));   %绘制功率谱曲线
xlabel('频率/Hz');ylabel('功率谱/dB');
title('平均周期图(重叠一半)N=1024');
grid on;
```

程序运行结果如图 9-9 所示。图 9-9（a）采用不重叠分段法的功率谱估计，图 9-9（b）为 2∶1 重叠分段的功率谱估计，可见后者估计曲线较为平滑。与上例比较，平均周期图法功率谱估计具有明显效果（涨落曲线靠近 0dB）。

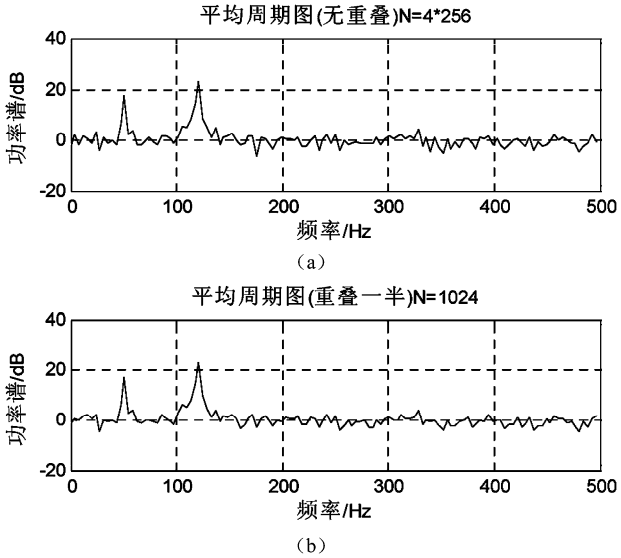


图 9-9　重叠和不重叠分段计算的功率谱比较

2．Welch 法

前面已经知道，把采样数据分段进行估计能减少周期图的方差，从而改进功率谱估计的直接法，即 Bartlett 法，但是，信号长度限制了可分的段数，如果要增加段数，可以使数据段之间的部分重叠，但这样会致使每一段的方差增大，所以对于需要再分段的段数的重叠率应加以平衡化。

提高周期图法估计的另一种方法就是采用对采样数据分段使用非矩形窗，即 Welch 法。由于非矩形窗在边沿趋近于零，从而减小了分段对重叠的依赖。选择合适的窗函数，采用每段一半的重叠率，能大大降低谱估计的方法。在这种方法中，记录数据仍分成 $K=\dfrac{N}{M}$ 段，即

$$x^{(i)}(n)=x(n+iM-N)\quad 0\leqslant n\leqslant M-1,1\leqslant i\leqslant K \tag{9-23}$$

每段 M 个取样，数据窗 $w(n)$ 在计算周期图之前就与数据段相乘，于是可定义 K 个修正周期图，即

$$J_M^{(i)}=\frac{1}{MU}\left|\sum_{n=0}^{M-1}x^{(i)}(n)w(n)\mathrm{e}^{-\mathrm{j}\omega n}\right|^2,\quad i=1,2,\cdots,K \tag{9-24}$$

U 是窗口序列函数的平均能量，即

$$U=\frac{1}{M}\sum_{n=0}^{M-1}w^2(n) \tag{9-25}$$

则定义谱估计为

$$B_x^w(\omega)=\frac{1}{K}\sum_{i=1}^{K}J_M^{(i)}(\omega) \tag{9-26}$$

【例 9-11】用程序实现 Welch 方法的功率谱估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=1000;          %采样频率
```

```
NFFT=256;
t=0:1/Fs:1;   %采样时间
x=sin(2*pi*100*t)+sin(2*pi*200*t)+sin(2*pi*400*t)+randn(size(t));
w=hanning(NFFT)';              %256 个点的海宁窗,注意取转置
pxx=(abs(fft(w.*x(1:NFFT))).^2+...
      abs(fft(w.*x(NFFT*1/2+1:NFFT*3/2))).^2+...
      abs(fft(w.*x(NFFT*2/2+1:NFFT*4/2))).^2+...
      abs(fft(w.*x(NFFT*3/2+1:NFFT*5/2))).^2+...
      abs(fft(w.*x(NFFT*4/2+1:NFFT*6/2))).^2+...
      abs(fft(w.*x(NFFT*5/2+1:NFFT*7/2))).^2)/(norm(w)^2*6);
f=(0:(NFFT-1))./NFFT*Fs;              %横坐标变换
PXX=10*log10(pxx);                    %纵坐标变换
figure
plot(f,PXX);
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
grid on;
```

运行程序，效果如图 9-10 所示。

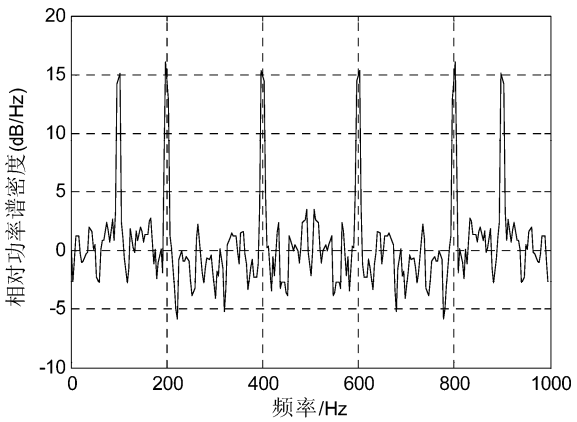


图 9-10　Welch 方法的功率谱估计

在 MATLAB 信号处理工具箱中，函数 psd 与 pwelch 都可以实现 Welch 法的功率谱估计。关于 psd 函数，前面已经介绍了，下面只对 pwelch 函数进行介绍。

功能：实现 Welch 法的功率谱估计，其调用格式如下：

```
[Pxx,w] = pwelch(x)
[Pxx,w] = pwelch(x,window)
[Pxx,w] = pwelch(x,window,noverlap)
[Pxx,w] = pwelch(x,window,noverlap,nfft)
[Pxx,w] = pwelch(x,window,noverlap,w)
[Pxx,f] = pwelch(x,window,noverlap,nfft,fs)
[Pxx,f] = pwelch(x,window,noverlap,f,fs)
[…] = pwelch(x,window,noverlap,…,'range')
pwelch(x,…)
```

【例 9-12】用 pwelch 函数实现 Welch 法的功率谱估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',0 );              %设置噪声的初始状态
Fs = 1000;                      %采样频率
t = 0:1/Fs:.3;                  %时间序列
%  输入信号
x = cos(2*pi*t*200) + randn(size(t));
pwelch(x,33,32,[],Fs,'twosided');
xlabel('频率/Hz');
title('利用 pwelch 函数实现功率谱估计');
```

运行程序，效果如图 9-11 所示。

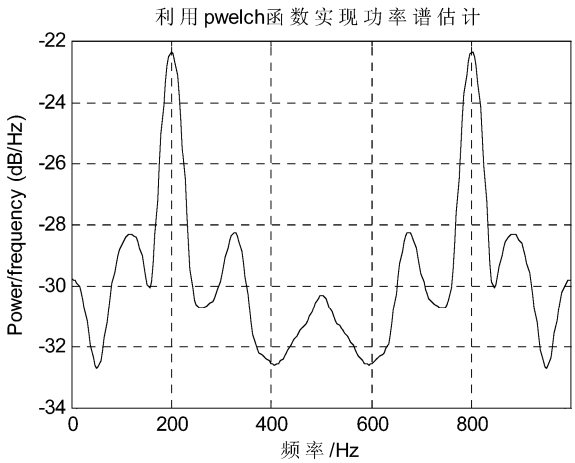


图 9-11　用 pwelch 函数实现功率谱估计

【例 9-13】对例 9-5 中的信号序列，用函数 psd 绘制自功率谱密度估计曲线。

其实现的 MATLAB 程序代码如下：

```
clear all;
Fs=1000;                        %采样频率
N=1024;Nfft=256;                %数据长度
n=0:N-1;t=n/Fs;                 %时间序列
window=hanning(256);            %选择的窗口
noverlap=128;                   %分段序列重叠的采样点数(长度)
dflag='none';                   %不进行趋势处理
randn('state',0);               %设置产生随机数的初始状态
%  带噪声的原始信号
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);
Pxx=psd(xn,Nfft,Fs,window,noverlap,dflag);   %功率谱估计
f=(0:Nfft/2)*Fs/Nfft;           %求得对应的频率向量
plot(f,10*log10(Pxx));          %绘制功率谱
xlabel('频率/Hz');ylabel('功率谱/dB');
title('PSD-Welch 方法');
grid on;
```

运行程序，效果如图 9-12 所示。可以看到，采用 Welch 法求得的功率谱与前面所述方法相比较，效果最好，使信号得以突出，抑制了噪声。

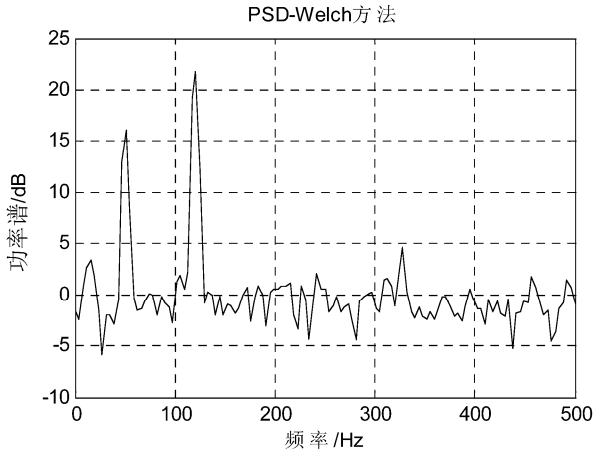


图 9-12　用 psd 实现 Welch 法的功率谱估计

【例 9-14】用 Welch 方法进行 PSD 估计，并比较当采用不同窗函数时的结果。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=1000;                %采样频率
NFFT=1024;
t=0:1/Fs:1;             %时间序列
x=sin(2*pi*100*t)+sin(2*pi*200*t)+sin(2*pi*400*t)+randn(size(t));   %信号
window1=boxcar(100);
window2=hamming(100);
window3=blackman(100);
window4=hanning(100);
noverlap=20;            %指定段与段之间的重叠的样本数
[pxx1,f1]=pwelch(x,window1,noverlap,NFFT,Fs);
[pxx2,f2]=pwelch(x,window2,noverlap,NFFT,Fs);
[pxx3,f3]=pwelch(x,window3,noverlap,NFFT,Fs);
[pxx4,f4]=pwelch(x,window4,noverlap,NFFT,Fs);
pxx1=10*log10(pxx1);
pxx2=10*log10(pxx2);
pxx3=10*log10(pxx3);
pxx4=10*log10(pxx4);
subplot(2,2,1);plot(f1,pxx1);
title('矩形窗');
subplot(2,2,2);plot(f2,pxx2);
title('海明窗');
subplot(2,2,3);plot(f3,pxx3);
title('布莱克曼窗');
subplot(2,2,4);plot(f4,pxx4);
title('汉宁窗');
```

运行程序，效果如图 9-13 所示。

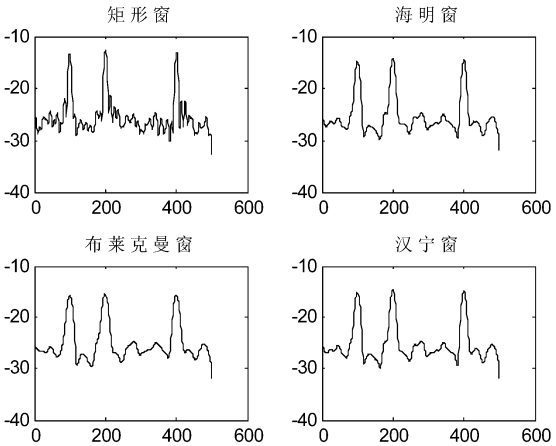


图 9-13　用 Welch 方法进行 PSD 估计，采用不同窗函数时的效果

9.2.3　AR 模型功率谱估计

传统的功率谱估计方法是利用加窗的数据或加窗的相关函数估计值的傅里叶变换来计算的，具有一定的优势，如计算效率高、估计值正比于正弦波信号的功率等，但是同时也存在许多缺点，主要缺点就是方差性能较差、谱分辨率低。而参数模型法可以大大提高功率谱估计的分辨率，是现代谱估计的主要研究内容，在语音分析、数据压缩以及通信等邻域有着广泛的应用。

按照模型化进行功率谱估计，其主要思路如下：

（1）选择模型；

（2）从给出的数据样本估计假设模型的参数；

（3）将估计出的模型参数代入模型的理论功率谱密度公式，得出一个较好的谱估计值。

下面，就对 AR 模型进行说明。

假设产生随机序列 $x(n)$ 的系统模型为一个线性差分方程，即

$$x(n)=\sum_{i=0}^{q}b_i w(n-i)-\sum_{j=0}^{q}a_j x(n-j) \tag{9-27}$$

式中，$w(n)$ 表示白噪声序列，对上式进行 Z 变换，可得

$$\sum_{j=0}^{q}a_j X(z)z^{-j}=\sum_{i=0}^{q}b_i W(z)z^{-i} \tag{9-28}$$

所以，系统的传递函数为

$$H(z)=\frac{X(z)}{W(z)}=\frac{B(z)}{A(z)} \tag{9-29}$$

式中

$$A(z)=\sum_{j=0}^{q}a_j z^{-j} \tag{9-30}$$

$$B(z)=\sum_{i=0}^{q}b_i z^{-i} \tag{9-31}$$

假定输入白噪声功率谱密度为 $P_w(z)=\sigma_w^2$，那么输出功率谱密度为

$$P_x(z)=\sigma_w^2\frac{B(z)B(z^{-1})}{A(z)A(z^{-1})} \tag{9-32}$$

又根据 $z=\mathrm{e}^{\mathrm{j}\omega}$，可得

$$P_w(z)=\sigma_w^2\left|\frac{B(\mathrm{e}^{\mathrm{j}\omega})}{A(\mathrm{e}^{\mathrm{j}\omega})}\right|^2 \tag{9-33}$$

这样，当确定了系数 a_j，b_i 和 σ_w^2 后，就可以求解得到随机信号的功率谱密度 $P_x(\omega)$。由式（9-33）可知，当 $i>0$，$b_i=0$ 时，系统的差分方程可变为

$$x(n)=-\sum_{j=1}^{q}a_j x(n-j)+w(n) \tag{9-34}$$

式（9-34）即为自回归模型，简称 AR（Auto-Regressive）模型，再对该式进行 Z 变换，得

$$H(z)=\frac{X(z)}{W(z)}=\frac{1}{A(z)}=\frac{1}{1+\sum_{j=1}^{q}a_j z^{-j}} \tag{9-35}$$

所以，AR 模型又称为全极点模型，AR 模型的输出功率谱为

$$P_x(\omega)=\frac{\sigma_w^2}{\left|A(\mathrm{e}^{\mathrm{j}\omega})\right|^2}=\frac{\sigma_w^2}{1+\left|\sum_{k=1}^{q}a_k\mathrm{e}^{-\mathrm{j}\omega k}\right|^2} \tag{9-36}$$

显然，计算 σ_w^2 和 a_k 后，就可以求解得到随机信号的功率谱密度 $P_x(\omega)$。同样，根据系数实际情况的不同，还可以得到 MA（Moving Average）模型和 ARMA 模型。

下面介绍围绕 AR 模型的几种谱估计方法。

1．Yule-Walker 法估计

通过模型分析法来进行功率谱估计，关键是要解决模型的参数估计问题。Yule-Walker 法又称为自相关法，其核心是从随机信号序列的自相关序列中计算出指定阶数的 AR 模型的参数，以得到该随机信号序列的功率谱估计，这种方法是用自相关法求解 AR 模型的参数。Yule-Walker 法估计通过如下的方程求解获得。Yule-Walker 方程求解可以用递推算法 Levinson-Durbin 实现：

$$\begin{bmatrix} r(1) & r^*(2) & \cdots & r^*(n) \\ r(2) & r(1) & \cdots & r^*(n-1) \\ \vdots & \vdots & & \vdots \\ r(n) & r(n-1) & \cdots & r(1) \end{bmatrix}\begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(n+1) \end{bmatrix}=\begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(n+1) \end{bmatrix} \tag{9-37}$$

式中：$a(2),\cdots,a(n+1)$ 是自回归系数；$r(1),r(2),\cdots,r(n+1)$ 为相关系数。

Yule-Walker 法 PSD 估计的公式为

$$\hat{P}_{\mathrm{Yulear}}=\frac{1}{\left|a^H e(f)\right|^2} \tag{9-38}$$

式中，$e(f)$ 为复数正弦曲线。

在 MATLAB 信号处理工具箱中，函数 pyulear 用来实现 Yule-walker AR 法的功率谱估计，其调用格式如下：

```
Pxx = pyulear(x,p)
Pxx = pyulear(x,p,nfft)
[Pxx,w] = pyulear(...)
[Pxx,w] = pyulear(x,p,w)
Pxx = pyulear(x,p,nfft,fs)
Pxx = pyulear(x,p,f,fs)
[Pxx,f] = pyulear(x,p,nfft,fs)
[Pxx,f] = pyulear(x,p,f,fs)
[Pxx,f] = pyulear(x,p,nfft,fs,'range')
[Pxx,w] = pyulear(x,p,nfft,'range')
pyulear(...)
```

【例 9-15】用 Yule-Walker AR 法进行 PSD 估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a = [1 -2.2137 2.9403 -2.1697 0.9606];   % AR 模型
% AR 模型频率响应
figure;freqz(1,a)
title('AR 系统频率响应');
randn('state',1);
x = filter(1,a,randn(256,1));       %  输出 AR 模型
figure;pyulear(x,4) ;
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
title('用 Yule-Walker AR 法进行谱估计');
grid on
```

运行程序，效果如图 9-14 及图 9-15 所示。

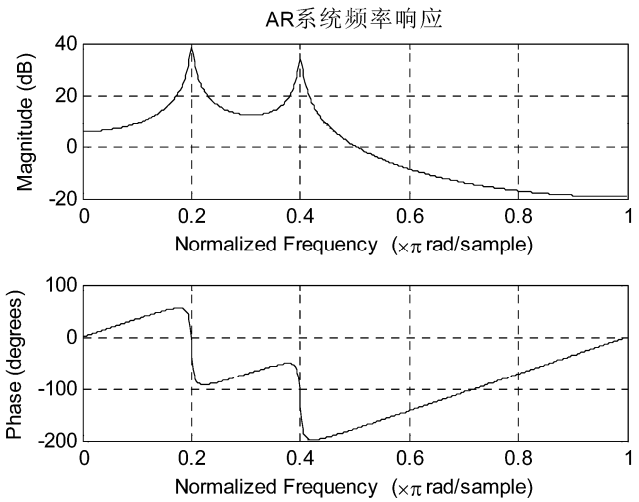


图 9-14　频率效果

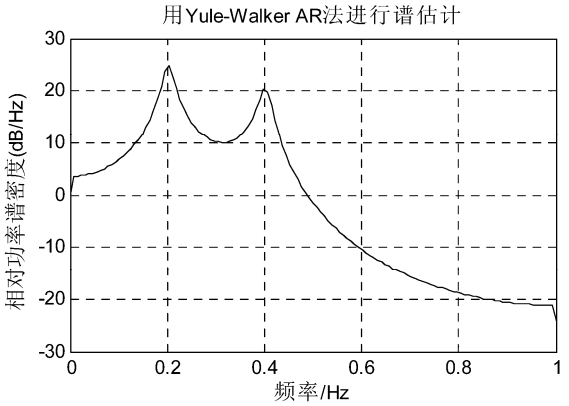


图 9-15　用 Yule-Walker AR 法进行谱估计

2．Burg 法估计

Burg 法是一种在 Levison-Durbin 递归约束的前提下，使前向和后向预测误差能量之和为最小的自回归功率谱估计的方法。Burg 方法避开了自相关函数的计算，它能够在低噪声的信号中分辨出非常接近的正弦信号，并且可以用较少的数据记录来进行估计，估计的结果非常接近真实值，而且，用 Burg 法得到的预测误差滤波器是最小相位的。但是，当用 Burg 法处理高阶模型、长数据记录时，结果的精度不是很高，并且有可能会出现谱线偏移和谱线分裂现象。

假定线性预测 AR 模型的前向预测误差和后向预测误差为 $f_p(n)$ 和 $b_p(n)$：

$$f_p(n)=x(n)+a_{p1}x(n-1)+\cdots+a_{pp}(n-p) \tag{9-39}$$

$$b_p(n)=x(n-p)+a_{p1}x(n-p+1)+\cdots+a_{pp}x(n) \tag{9-40}$$

前后预测误差的功率之和为

$$P_{fb}=\frac{1}{2}[P_f+P_b]=\frac{1}{N-p}\sum_{n=p}^{N-1}\left|f_p(n)\right|^2+\frac{1}{N-p}\sum_{n=p}^{N-1}\left|b_p(n)\right|^2 \tag{9-41}$$

$f_p(n)$ 和 $b_p(n)$ 存在下面的递推关系：

$$\begin{aligned} f_s(n)&=f_{s-1}(n)+h_s b_{s-1}(n-1)\\ b_s(n)&=b_{s-1}(n)+h_s f_{s-1}(n-1)\end{aligned} \tag{9-42}$$

式中，s 为阶次，$s=1,2,\cdots,p$；h_s 为反射系数，且有 $h_s=a_{ss}$，而且

$$f_0(n)=b_0(n)=x(n) \tag{9-43}$$

根据 Burg 法使得前向和后向预测误差能量之和相对于反射系数为最小，可得 h_s 的估计公式为

$$\hat{h}_s=\frac{-2\sum_{n=s}^{N-1}f_{s-1}(n)b_{s-1}(n-1)}{\sum_{n=s}^{N-1}\left|f_s(n)\right|^2+\sum_{n=s}^{N-1}\left|b_{s-1}(n-1)\right|^2} \tag{9-44}$$

然后，便可以由 Levinson-Durbin 递推算法求出 s 阶次的 AR 模型的参数：

$$
\begin{aligned}
&a_{s,i} = a_{s-1,i} + \hat{h}_s a_{s-1,s-i} \\
&a_{ss} = \hat{h}_s \\
&\sigma_s^2 = (1-|h_s|^2)\sigma_{s-1}^2, \quad \sigma_0^2 = \hat{R}_x(0) = \frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2
\end{aligned}
\tag{9-45}
$$

在 MATLAB 信号处理工具箱中，函数 arburg 用上述的 Burg 算法计算 AR 模型的参数，而 pburg 用来实现 Burg AR 法的功率谱估计。

1）arburg 函数

功能：利用 Burg 算法计算 AR 模型参数。其调用格式如下：

a = arburg(x,p)

[a,e] = arburg(x,p)

[a,e,k] = arburg(x,p)

【例 9-16】用 Burg 算法计算 AR 模型的参数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('seed',0);
a=[1 0.1 0.2 0.3 0.4 0.5 0.6];
x=impz(1,a,20)+randn(20,1)/20;
[A,E,K]=arburg(x,5)
```

运行程序，输出如下：

```
A =
    1.0000   -0.2505    0.1053    0.2217    0.2361    0.4025
E =
    0.0633
K =
   -0.5574
    0.3562
    0.4531
    0.4021
    0.4025
```

2）pburg 函数

功能：实现 Burg AR 的功率谱估计。其调用格式如下：

Pxx = pburg(x,p)

Pxx = pburg(x,p,nfft)

[Pxx,w] = pburg(...)

[Pxx,w] = pburg(x,p,w)

Pxx = pburg(x,p,nfft,fs)

Pxx = pburg(x,p,f,fs)

[Pxx,f] = pburg(x,p,nfft,fs)

[Pxx,f] = pburg(x,p,f,fs)

[Pxx,f] = pburg(x,p,nfft,fs,'range')

[Pxx,w] = pburg(x,p,nfft,'range')

pburg(...)

【例 9-17】用 Burg 法进行 PSD 估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
a = [1 -2.2137 2.9403 -2.1697 0.9606];          %定义 AR 模型
[H,w] = freqz(1,a,256);                         % AR  模型的频率响应
Hp = plot(w/pi,20*log10(2*abs(H)/(2*pi)),'r');
hold on;
randn('state',1);
x = filter(1,a,randn(256,1));                   % AR  模型输出
pburg(x,4,511);
xlabel('频率/Hz')
ylabel('相对功率谱密度(dB/Hz)');
title('用 Burg 法进行 PSD 估计');
legend('PSD 模型输出','PSD 谱估计');
grid on;
```

运行程序，效果如图 9-16 所示。

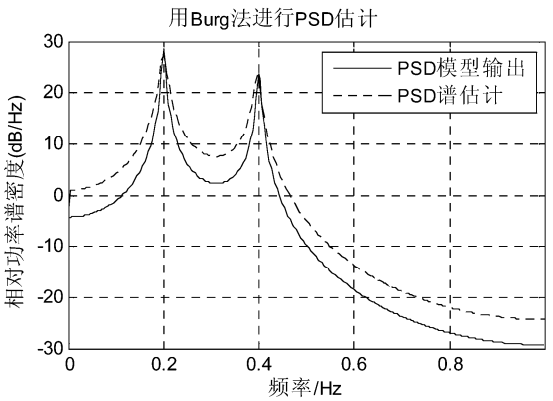


图 9-16　Burg 法 PSD 估计效果

3．协方差法估计

自回归功率谱估计的协方差法是一种基于使前向预测误差最小的技术，而改进的协方差方法则是同时使前向和后向预测误差均最小的技术。在 MATLAB 信号处理工具箱中，函数 pcov 用来实现自回归功率谱估计的协方差方法，而函数 pmcov 用来实现自回归功率谱估计的改进的协方差方法。

这两个函数的具体调用格式与前述的 pyulear 函数和 pburg 函数大致相同，这里不再给出其调用格式。

【例 9-18】比较协方差方法与改进的协方差方法在噪声信号的功率谱估计中的效果。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
fs=1000;                    %采样频率
h=fir1(20,0.3);
r=randn(1024,1);            %加入的噪声
x=filter(h,1,r);
[pxx1,f]=pcov(x,20,[],fs);
```

```
[pxx2,f]=pmcov(x,20,[],fs);
pxx1=10*log10(pxx1);
pxx2=10*log10(pxx2);
plot(f,pxx1,'r:',f,pxx2,'s');
ylabel('相对幅度/dB');xlabel('功率谱估计');
title('协方差估计法');
legend('协方差法','改进的协方差法');
```

运行程序，效果如图 9-17 所示。

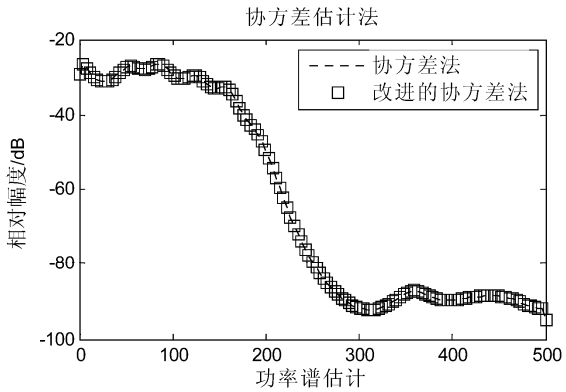


图 9-17　协方差法与改进的协方差法在噪声信号的功率谱估计中的效果

9.2.4　部分现代谱估计的非参数方法

在功率谱的现代谱估计方法中，除了前面所讲的参数模型功率谱估计以外，还有一类方法，这就是现代谱估计中的非参数方法。

1．MTM（Multitaper）估计法

MTM 法使用正交的窗口来截取获得相互独立的功率谱估计，然后再把这些估计结果综合得到最终的估计。MTM 法最重要的参数是时间与带宽的乘积——NW。此参数直接影响到谱估计的窗的个数，其中窗的个数为 2*NW−1 个。因此，随着 NW 的增大，窗的个数增多，会有更多的谱估计，从而谱估计的方差得到减小。但是，同时会带来谱泄漏的增大，而且正的谱估计的结果将会有更大的偏差。因此，在使用本方法估计功率谱的时候，就存在一个 NW 的选择问题，应尽量保证在偏差和方差之间取得最大的平衡。

MTM 估计法，显示了更多的自由度，并且比较容易给出估计期望值偏差与方差间权衡的定量算法，而且，在 MTM 法中，增加的窗可以用于复原一些丢失的信息。

在 MATLAB 信号处理工具箱中提供了 pmtm 函数，实现 Multitaper 法的功率谱估计，其调用格式如下：

```
[Pxx,w] = pmtm(x,nw)
[Pxx,w] = pmtm(x,nw,nfft)
[Pxx,w] = pmtm(x,nw,w)
[Pxx,f] = pmtm(x,nw,nfft,fs)
[Pxx,w] = pmtm(x,nw,f,fs)
[Pxx,Pxxc,f] = pmtm(x,nw,nfft,fs)
[Pxx,Pxxc,f] = pmtm(x,nw,nfft,fs,p)
```

```
[Pxx,Pxxc,f] = pmtm(x,e,v,nfft,fs,p)
[Pxx,Pxxc,f] = pmtm(x,dpss_params,nfft,fs,p)
[...] = pmtm(...,'DropLastTaper',dropflag)
[...] = pmtm(...,'method')
[...] = pmtm(...,'range')
pmtm(...)
```

【例 9-19】在置信度为 99%区间上利用 MTM 法估计有色噪声。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',0);
fs = 1000;
t = 0:1/fs:0.3;
x = cos(2*pi*t*200) + 0.1*randn(size(t));
[Pxx,Pxxc,f] = pmtm(x,3.5,512,fs,0.99);
hpsd = dspdata.psd([Pxx Pxxc],'Fs',fs);
plot(hpsd)
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
title('MTM 法估计');
grid on;
```

运行程序，效果如图 9-18 所示。

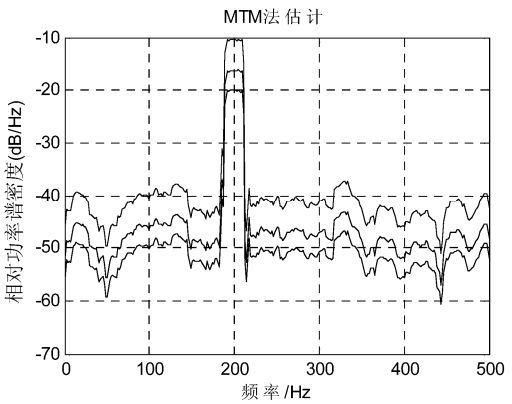


图 9-18　MTM 法估计频谱

【例 9-20】用多窗口法（MTM），分别采用 NW=4 和 NW=2，估计例 6-24 的含有噪声和 50Hz、120Hz 周期信号的功率谱密度。

其实现的 MATLAB 程序代码如下：

```
clear all;
Fs=1000;                                %采样频率
N=1024;Nfft=256;                        %数据长度
n=0:N-1;t=n/Fs;                         %分段数据长度,时间序列
randn('state',0);                       %设置产生随机数的初始状态
%  带噪声的原始信号
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);
[Pxx1,f]=pmtm(xn,4,Nfft,Fs);            %用多窗口法(NW=4)估计功率谱
subplot(211);plot(f,10*log10(Pxx1));    %绘制功率谱
```

```
xlabel('频率/Hz');ylabel('功率谱/dB');
title('多窗口法(MTM) NW=4');
grid on;
[Pxx2,f]=pmtm(xn,2,Nfft,Fs);              %用多窗口法(NW=2)估计功率谱
subplot(212);plot(f,10*log10(Pxx2));      %绘制功率谱
xlabel('频率/Hz');ylabel('功率谱/dB');
title('多窗口法(MTM) NW=2');
grid on;
```

运行程序，效果如图 9-19 所示。

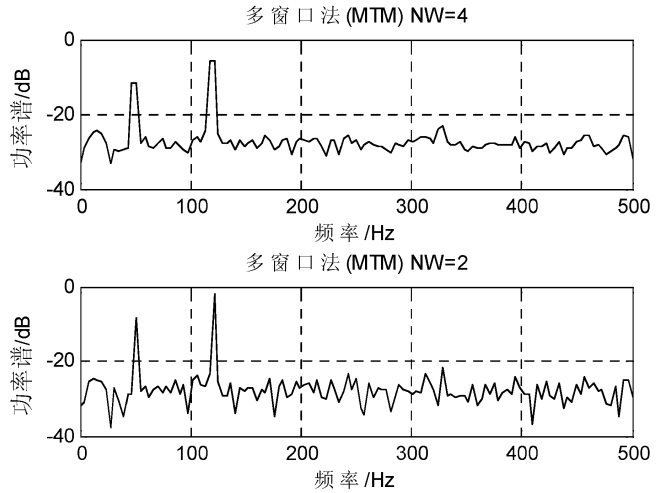


图 9-19　MTM 显示频谱效果

由图 9-19 可以看到，NW=4 和 2 均得到了比较好的功率谱估计。图 9-19（a）中 NW=4 估计的功率谱采用较多的窗口，使得每个窗口的数据点数减少，因此频率域分辨率降低，功率谱的波动较少。图 9-19（b）为 NW=2 的情况，采用较少的窗口，每个窗口的数据点增多，使得频率域的分辨率提高，但噪声的分辨率也随之增高，使得功率谱具有相对较大的波动。

2．特征向量（AV）法估计

定义信号向量 $\boldsymbol{e}_i=[1,\exp(\mathrm{j}\omega_i),\cdots,\exp(\mathrm{j}\omega_i p)]^{\mathrm{T}}$，$i=1,2,\cdots,M$，则

$$\boldsymbol{R}_{p+1}=\sum_{i=1}^{M}\boldsymbol{A}_i\boldsymbol{e}_i\boldsymbol{e}_i^{\mathrm{H}}+\sigma^2\boldsymbol{I}_{p+1} \tag{9-46}$$

令 $\boldsymbol{S}_{p+1}=\sum_{i=1}^{M}\boldsymbol{A}_i\boldsymbol{e}_i\boldsymbol{e}_i^{\mathrm{H}}$，对 $\boldsymbol{S}_{p+1}$ 进行特征分解，得

$$\boldsymbol{S}_{p+1}=\sum_{i=1}^{p+1}\lambda_i\boldsymbol{V}_i\boldsymbol{V}_i^{\mathrm{H}} \tag{9-47}$$

$\boldsymbol{V}_i$ 是对应于特征值 λ_i 的特征向量，且它们之间是相互正交的，即

$$\boldsymbol{V}_i\boldsymbol{V}_i^{H}=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases}\quad i,j=1,2,\cdots,M \tag{9-48}$$

单位矩阵 $\boldsymbol{I}_{p+1}$ 也可用特征向量 $\boldsymbol{V}_i$ 表示为

$$\boldsymbol{I}_{p+1}=\sum_{i=1}^{p+1}\boldsymbol{V}_i\boldsymbol{V}_i^{\mathrm{H}} \tag{9-49}$$

可以证明，$\boldsymbol{S}_{p+1}$ 的秩最大为 M，若 M 小于 $p+1$，那么 $\boldsymbol{S}_{p+1}$ 将有 $p+1-M$ 个零特征值，若将特征值按大小次序排列，则 $\boldsymbol{S}_{p+1}$ 的特征分解可写成

$$\boldsymbol{S}_{p+1}=\sum_{i=1}^{M}\lambda_i\boldsymbol{V}_i\boldsymbol{V}_i^{\mathrm{H}} \tag{9-50}$$

$\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_M$ 称为主特征向量。

由上面的分析，得

$$\boldsymbol{R}_{p+1}=\sum_{i=1}^{M}(\lambda_i+\sigma^2)\boldsymbol{V}_i\boldsymbol{V}_i^{\mathrm{H}}+\sum_{i=M+1}^{p+1}\sigma^2\boldsymbol{V}_i\boldsymbol{V}_i^{\mathrm{H}} \tag{9-51}$$

式（9-51）即为相关矩阵的特征分解。显然，$\boldsymbol{R}_{p+1}$ 和信号矩阵 $\boldsymbol{S}_{p+1}$ 有着相同的特征向量，它们的所有特征向量 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_{p+1}$ 形成了一个 $p+1$ 维的向量空间，且它们是相互正交的。进一步，该向量空间又可分成两个子空间：一个是由特征向量 $\boldsymbol{V}_{M+1},\boldsymbol{V}_{M+2},\cdots,\boldsymbol{V}_{p+1}$ 形成的噪声空间，每个向量的特征值都是 σ^2；另一个是由主特征向量 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_M$ 形成的信号空间，其特征值分别是 $(\lambda_1+\sigma^2),(\lambda_2+\sigma^2),\cdots,(\lambda_M+\sigma^2)$，$\sigma^2$ 在此反映了噪声对信号空间的影响。

由于信号向量 $\boldsymbol{e}_i$ 和噪声空间的各个向量 $\boldsymbol{V}_{M+1},\boldsymbol{V}_{M+2},\cdots,\boldsymbol{V}_{p+1}$ 都是正交的，因此，和它们的线性组合也是正交的，即

$$\boldsymbol{e}_i^{\mathrm{H}}\left(\sum_{i=M+1}^{p+1}a_k\boldsymbol{V}_k\right)=0,\quad i=1,2,\cdots,M \tag{9-52}$$

令 $\boldsymbol{e}(\omega)=[1,\exp(\mathrm{j}\omega),\cdots,\exp(\mathrm{j}\omega p)]^{\mathrm{T}}$，则有

$$\boldsymbol{e}^{\mathrm{H}}(\omega)\left(\sum_{i=M+1}^{p+1}a_k\boldsymbol{V}_k\boldsymbol{V}_k^{\mathrm{H}}\right)\boldsymbol{e}(\omega)=\sum_{i=M+1}^{p+1}a_k\left|\boldsymbol{e}^{\mathrm{H}}(\omega)\boldsymbol{V}_k\right|^2 \tag{9-53}$$

当 $\omega=\omega_i$ 时，应为零，有

$$\hat{\boldsymbol{P}}_x(\omega)=\frac{1}{\sum_{i=M+1}^{p+1}a_k\left|\boldsymbol{e}^{\mathrm{H}}(\omega)\boldsymbol{V}_k\right|^2} \tag{9-54}$$

在 $\omega=\omega_i$ 处，应是无穷大，但由于 $\boldsymbol{V}_k$ 是由相关矩阵分解而得，而相关矩阵是估计出来的，因此必有误差，所以 $\hat{\boldsymbol{P}}_x(\omega_i)$ 为有限值，但呈现尖的峰值，其峰值对应的频率即是正弦信号的频率，由此也可得到序列的功率谱估计，其功率谱的分辨率要好于 AR 模型。

（1）若令 $a_k=1$，$k=M+1,\cdots,p+1$，所得到估计即为 MUSIC 估计，即

$$\hat{\boldsymbol{P}}_{\mathrm{MUSIC}}(\omega)=\frac{1}{\boldsymbol{e}^{\mathrm{H}}(\omega)\left(\sum_{i=M+1}^{p+1}\boldsymbol{V}_k\boldsymbol{V}_k^{\mathrm{H}}\right)\boldsymbol{e}(\omega)} \tag{9-55}$$

（2）若令 $a_k=1/\lambda_k$，$k=M+1,\cdots,p+1$，所得功率谱称为特征向量估计，即

$$\hat{\boldsymbol{P}}_{\mathrm{EV}}(\omega)=\frac{1}{\boldsymbol{e}^{\mathrm{H}}(\omega)\left(\sum_{i=M+1}^{p+1}\frac{1}{\lambda_k}\boldsymbol{V}_k\boldsymbol{V}_k^{\mathrm{H}}\right)\boldsymbol{e}(\omega)} \tag{9-56}$$

在 MATLAB 信号处理工具箱中提供了 peig 函数，实现特征向量法的功率谱估计，其调用格式如下：

[S,w] = peig(x,p)
[S,w] = peig(x,p,w)
[S,w] = peig(...,nfft)
[S,f] = peig(x,p,nfft,fs)
[S,f] = peig(x,p,f,fs)
[S,f] = peig(...,'corr')
[S,f] = peig(x,p,nfft,fs,nwin,noverlap)
[...] = peig(...,'range')
[...,v,e] = peig(...)
peig(...)

【例 9-21】用特征向量法进行 PSD 估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('state',1);
n=0:99;
s=exp(i*pi/2*n)+2*exp(i*pi/4*n)+exp(i*pi/3*n)+randn(1,100);
X=corrmtx(s,12,'mod');
peig(X,3,'whole');
grid on;
xlabel('归一化频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
title('特征向量法进行 PSD 估计');
```

运行程序，效果如图 9-20 所示。

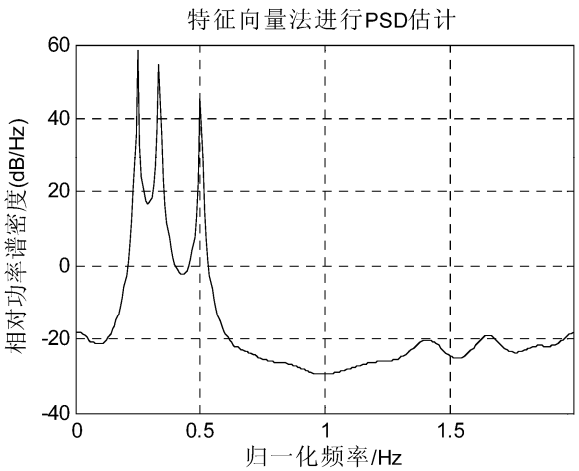


图 9-20　特征向量法进行 PSD 估计效果

9.3　MUSIC 法功率谱估计

由 $P(\omega)=\dfrac{1}{\boldsymbol{a}^{\mathrm{H}}(\omega)\boldsymbol{G}\boldsymbol{G}^{\mathrm{H}}\boldsymbol{a}(\omega)}$ 定义的函数 $P(\omega)$ 描述了空间参数（即波达方向）的分布，故称为空间谱。由于它能对多个空间信号进行识别，所以此方法也称为多信号分类法，简称 MUSIC

（Multiple Signal Classification）法。

MATLAB 信号处理工具箱还提供另一种功率谱估计函数 pmusic，该矩阵执行 MUSIC 法。将数据自相关矩阵看成由信号自相关矩阵和噪声自相关矩阵两部分组成，即数据自相关矩阵 ***R*** 包含两个子空间信息：信号子空间和噪声子空间。这样，矩阵特征值向量也可分为两个子空间：信号子空间和噪声子空间。为了求得功率谱估计，函数 pmusic 计算信号子空间和噪声子空间的特征值向量函数，使得在周期信号频率处函数值最大，功率谱估计出现峰值，而在其他频率处函数最小，其调用格式如下：

```
[S,w] = pmusic(x,p)
[S,w] = pmusic(x,p,w)
[S,w] = pmusic(...,nfft)
[S,f] = pmusic(x,p,nfft,fs)
[S,f] = pmusic(x,p,f,fs)
[S,f] = pmusic(...,'corr')
[S,f] = pmusic(x,p,nfft,fs,nwin,noverlap)
[...] = pmusic(...,'range')
[...,v,e] = pmusic(...)
pmusic(...)
```

【例 9-22】用多信号分类法，采用 7 个窗口，估计例 9-5 含有噪声和 50Hz、120Hz 周期信号的功率谱密度。

其实现的 MATLAB 程序代码如下：

```
clear all;
Fs=1000;                    %采样频率
N=1024;Nfft=256;            %数据长度
n=0:N-1;t=n/Fs;             %时间序列
randn('state',0);           %设置产生随机数的初始状态
%  带噪声的原始信号
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(1,N);
pmusic(xn,[7,1.1],Nfft,Fs,32,16);   %采用多信号分类法估计功率谱
xlabel('频率/Hz');ylabel('功率谱/dB');
title('通过 MUSIC 法估计的伪谱');
grid on;
```

运行程序，效果如图 9-21 所示。可见结果较为清楚地识别出信号中所含的频率成分，并且具有较高的分辨率。但要注意，函数 pmusic 参数的选择对估计的功率谱影响较大。

【例 9-23】设序列 x 有两个正弦信号组成，其频率分别为 f1=200Hz，f2=202Hz，采样频率为 Fs=1000Hz，并有含一定的白噪声，通过 peig 函数和 pmusic 函数进行谱估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
Fs=1000;                    %频率
t=0:1/Fs:1-1/Fs;            %时间序列
x=5*cos(2*pi*200*t)+5*cos(2*pi*202*t)+randn(1,length(t));
NFFT=1024;
p=40;
pxx=pmusic(x,p,NFFT,Fs);      %MUSIC 估计
```

```
k=0:floor(NFFT/2-1);
figure;
subplot(2,1,1);plot(k*Fs/NFFT,10*log10(pxx(k+1)));
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
title('MUSIC 法谱估计');
pxx1=peig(x,p,NFFT,Fs);     %特征向量估计
k=0:floor(NFFT/2-1);
subplot(2,1,2);plot(k*Fs/NFFT,10*log10(pxx1(k+1)));
xlabel('频率/Hz');ylabel('相对功率谱密度(dB/Hz)');
title('特征向量法谱估计');
```

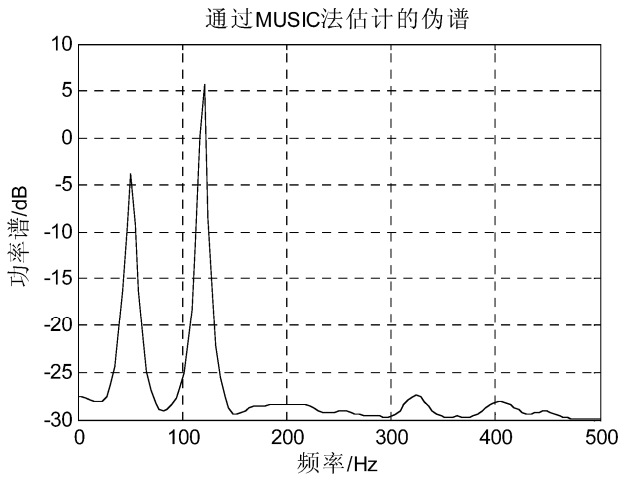


图 9-21　多信号分类法估计的功率谱

运行程序，效果如图 9-22 所示。

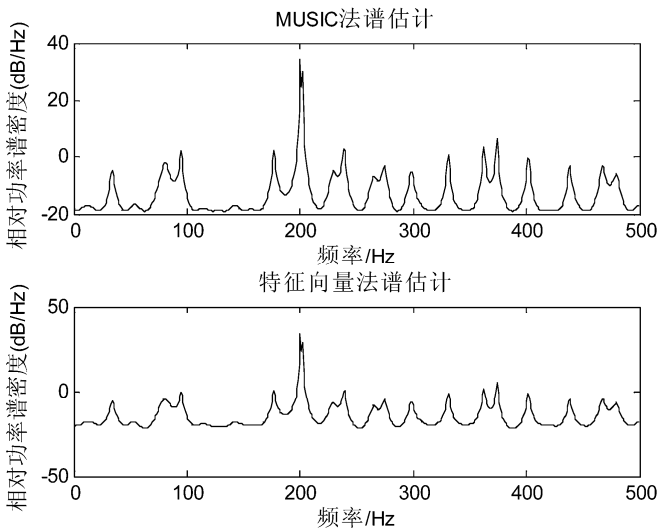


图 9-22　peig 函数和 pmuisc 函数谱估计效果图

9.4 相干函数分析

两个信号 $x(t)$ 和 $y(t)$ 之间的相干函数为

$$C_{xy}(\omega)=\frac{\left|P_{xy}(\omega)\right|^2}{P_{xx}(\omega)P_{yy}(\omega)} \tag{9-57}$$

式中，$P_{xy}(\omega)$ 为 $x(t)$ 和 $y(t)$ 的互功率谱密度，$P_{xx}(\omega)$ 和 $P_{yy}(\omega)$ 分别为 $x(t)$ 和 $y(t)$ 的自功率谱密度。

相干函数 $C_{xy}(\omega)$ 为 0～1 的实数，它用来检测信号 $x(t)$ 和 $y(t)$ 在频域内的相关程度。若 $y(t)$ 为 $x(t)$ 的线性响应，则 $C_{xy}(\omega)=1$；若 $x(t)$ 和 $y(t)$ 完全互不相关，则 $C_{xy}(\omega)=0$。通常在测试过程中，$0<C_{xy}(\omega)<1$，这表明有三种可能：

（1）联系 $x(t)$ 和 $y(t)$ 的系统不完全是线性的。

（2）系统输出 $y(t)$ 是由 $x(t)$ 和其他信号共同输入引起的。

（3）在输出端有噪声干扰混入。

由于相干函数 $C_{xy}(\omega)$ 的上述特点使它在工程上得到一些应用，如检查两个信号之间的因果关系。

【例 9-24】（1）设计一个归一化截止频率为 0.2 的 FIR 滤波器，将带有白噪声的频率为 50Hz 的周期信号输入滤波器，求输入信号和输出信号的相干函数；（2）计算两个独立信号 x(t)=u(t)，y(t)=sin(2*pi*50*t)的相干函数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%第一种情况:y 与 x 相干
Fs=1000;                          %采样频率
N=1024;                           %数据长度
NFFT=256;                         %分段数据长度
n=0:N-1; t=n/Fs;                  %时间序列
window=hanning(256);              %采用的窗口
noverlap=128;                     %重叠数据个数
dflag='none';
randn('state',0);                 %设置产生随机数的状态
xn=sin(2*pi*50*t)+randn(1,N);     %原始信号
h=fir1(30,0.2,boxcar(31));        %设置 30 阶截止频率为 0.2 的 FIR 滤波器
yn=filter(h,1,xn);                %运用上述的 FIR 滤波器进行滤波
subplot(2,1,1);cohere(xn,yn,NFFT,Fs,window,noverlap,dflag);  %计算其相关函数
xlabel('频率/Hz');ylabel('相干函数估计');
title('Yn 与 Xn 相干');
grid on;
% 第二种情况,y 与 x 不相干
Fs=1000;                          %采样频率
N=1024;                           %数据长度
NFFT=256;                         %分段数据长度
n=0:N-1; t=n/Fs;                  %时间序列
```

```
window=hanning(256);                    %采用的窗口
noverlap=128;                           %重叠数据个数
dflag='none';
randn('state',0);                       %设置产生随机数的状态
xn=ones(1,N);                           %第一个信号
yn=sin(2*pi*50*t);                      %第二个信号
subplot(2,1,2);cohere(xn,yn,NFFT,Fs,window,noverlap,dflag);   %两个信号的相干函数
xlabel('频率/Hz');ylabel('相干函数估计');
title('Yn 与 Xn 不相干');
grid on;
```

运行程序，效果如图 9-23 所示。

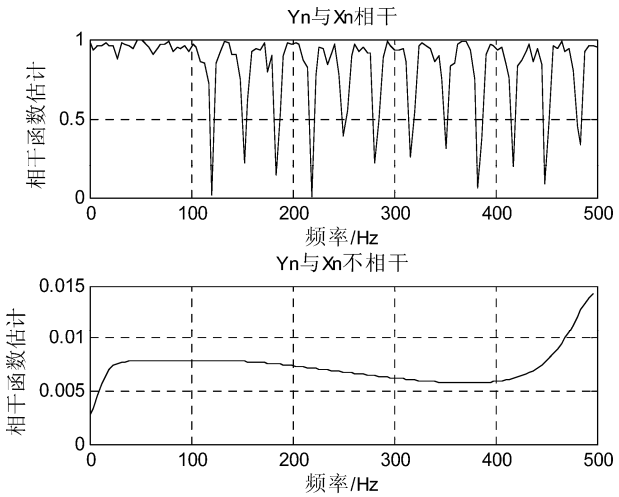


图 9-23　滤波器输入和输出的相干函数与互不相干序列的相干函数的比较效果

9.5　参数建模

随机信号的傅里叶变换是不存在的，因此无法像确定性信号那样用数学表达式来精确地描述它，而只能用它的各种统计平均量来表征，其中自相关函数最能完整地表征它的特定统计平均　量值。而一个随机信号的功率谱密度正是自相关函数的傅里叶变换，可以用功率谱密度来表征它的统计平均谱特性。因此，要在统计意义下描述一个随机信号，就需要估计它的功率谱密度（PSD）。

9.5.1　参数建模的基本概念

参数模型法是现代谱估计的主要内容，它的主要思路如下：

（1）假定所研究的过程 $x(n)$ 是由一个输入序列 $u(n)$ 激励的线性系统 $H(z)$ 的输出。

（2）由已知的 $x(n)$，或其自相关函数 $C_{xx}(m)$ 来估计 $H(z)$ 的参数。

（3）由 $H(z)$ 的参数来估计 $x(n)$ 的功率谱。

对于一个确定性系统，输入信号 $u(n)$ 和输出信号 $x(n)$ 之间总有如下的关系：

$$x(n)=-\sum_{k=1}^{p}a_k x(n-k)+\sum_{k=0}^{q}b_k u(n-k) \qquad (9\text{-}58)$$

以及

$$x(n)=\sum_{k=0}^{\infty}h(k)u(n-k) \tag{9-59}$$

9.5.2 频率域建模

频率域建模就是由滤波器（系统）的频率响应建立其模型。根据频率域类型分为面向模拟滤波器的 s 域内建模和面向数字滤波器的 z 域内建模。

1．模拟滤波器的 s 域建模

模拟滤波器传递函数的一般形式为

$$H(s)=\frac{B(s)}{A(s)}=\frac{b(1)s^{nb}+b(2)s^{nb-1}+\cdots+b(nb+1)}{a(1)s^{na}+a(2)s^{na-1}+\cdots+a(na+1)} \tag{9-60}$$

MATLAB 信号处理工具箱提供了 invfreqs 函数，用于由给定的频率响应数据求式（9-60）所示的模拟滤波器传递函数，其调用格式如下：

```
[b,a] = invfreqs(h,w,n,m)
[b,a] = invfreqs(h,w,n,m,wt)
[b,a] = invfreqs(h,w,n,m,wt,iter)
[b,a] = invfreqs(h,w,n,m,wt,iter,tol)
[b,a] = invfreqs(h,w,n,m,wt,iter,tol,'trace')
[b,a] = invfreqs(h,w,'complex',n,m,...)
```

若已知复频率响应的幅值向量 mag 和相位向量 phase，在 MATLAB 中要采用 h=mag.*exp(j*phase)将其复合为复数形式。

【例 9-25】invfreqs 函数用法示例。

```
>> clear all;
a = [1 2 3 2 1 4]; b = [1 2 3 2 3];
[h,w] = freqs(b,a,64);
[bb,aa] = invfreqs(h,w,4,5)
bb =
    1.0000    2.0000    3.0000    2.0000    3.0000
aa =
    1.0000    2.0000    3.0000    2.0000    1.0000    4.0000
>> [bbb,aaa] = invfreqs(h,w,4,5,[],30)
bbb =
    0.6816    2.1015    2.6694    0.9113   -0.1218
aaa =
    1.0000    3.4676    7.4060    6.2102    2.5413    0.0001
```

2．数字滤波器的 z 域内建模

数字滤波器传递函数的一般形式为

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(nb+1)z^{-nb}}{a(1)+a(2)z^{-1}+\cdots+a(na+1)z^{-na}} \tag{9-61}$$

MATLAB 信号处理工具箱提供了 invfreqz 函数，用于由给定的频率响应数据求式（9-61）所示的数字滤波器的传递函数，其调用格式如下：

```
[b,a] = invfreqz(h,w,n,m)
```

[b,a] = invfreqz(h,w,n,m,wt)

[b,a] = invfreqz(h,w,n,m,wt,iter)

[b,a] = invfreqz(h,w,n,m,wt,iter,tol)

[b,a] = invfreqz(h,w,n,m,wt,iter,tol,'trace')

[b,a] = invfreqz(h,w,'complex',n,m,...)

【例 9-26】invfreqz 函数用法示例。

```
>> clear all;
a = [1 2 3 2 1 4]; b = [1 2 3 2 3];
[h,w] = freqz(b,a,64);
[bb,aa] = invfreqz(h,w,4,5)
bb =
    1.0000    2.0000    3.0000    2.0000    3.0000
aa =
    1.0000    2.0000    3.0000    2.0000    1.0000    4.0000
>> [bbb,aaa] = invfreqz(h,w,4,5,[],30)
bbb =
    0.2427    0.2788    0.0069    0.0971    0.1980
aaa =
    1.0000   -0.8944    0.6954    0.9997   -0.8933    0.6949
```

【例 9-27】已知原始滤波器为 4 阶巴特沃思低通数字滤波器，归一化频率为 0.4，用其频率响应数据产生一个 3 阶滤波器，试比较采用等误差和输出误差迭代算法的结果。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[b,a]=butter(4,0.4);                    %产生 4 阶巴特沃思滤波器
[h,w]=freqz(b,a,64);                    %计算频率响应
[bb,aa]=invfreqz(h,w,3,3);              %采用 invfreqz 求解滤波器的系数
wt=ones(size(w));                       %产生权向量
niter=30;                               %迭代次数
[bbb,aaa]=invfreqz(h,w,3,3,wt,niter);   %采用权向量和迭代次数求滤波器系数
[h1,w1]=freqz(bb,aa,w);                 %计算模型 1 的频率响应
[h2,w2]=freqz(bbb,aaa,w);               %计算模型 2 的频率响应
plot(w/pi,abs(h),'r',w1/pi,abs(h1),'k:',w2/pi,abs(h2),'b.');   %绘制频率响应
xlabel('归一化频率');ylabel('振幅');
legend('理想模型','未采用权向量和迭代次数','采用权向量和迭代次数');
grid on;
sumerr1=sum(abs(h-h1).^2)               %输出模型 1 误差
sumerr2=sum(abs(h-h2).^2)               %输出模型 2 误差
```

运行程序，输出如下，效果如图 9-24 所示。

```
sumerr1 =      0.0200
sumerr2 =      0.0096
```

程序运行结果表明，输出误差迭代法（预估算法）具有较好的拟合精度。

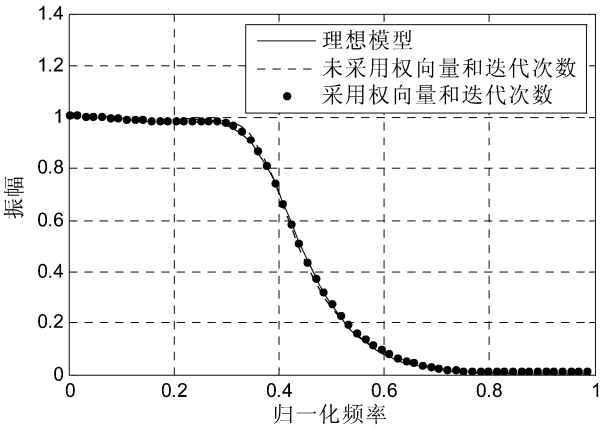


图 9-24　两种算法估计模型的幅频特性的比较效果

第 10 章　小波分析在信号处理中的应用

10.1　信号的小波变换

目前，小波变换在许多工程领域中都得到了广泛的应用，成为科技工作者经常使用的工具之一。

信号分析领域有许多处理方法和工具，其中最常用的是傅里叶分析，它将时域信号转换为频域信号，从而可以在频域对信号进行分析。但傅里叶分析在检测信号中包含的趋势、突变、事件的开始与结束等特征时显得无能为力，因为它不具备局部分析的能力。而小波分析是一个范围可变的窗口方法，小波分析可以用长的时间窗口来获得低频信息，用短的时间窗口来获得高频信息，因而小波分析具备局部分析与细化的能力。

10.1.1　信号的连续小波变换

小波是通过对基本小波进行尺度伸缩和位移得到的。基本小波是一个具有特殊性质的实值函数，其振荡快速衰减，且在数学上满足积分为零的条件，即

$$\int_{-\infty}^{\infty}\psi(t)\mathrm{d}t=0 \tag{10-1}$$

其频谱满足如下条件：

$$C_{\psi}=\int_{-\infty}^{\infty}\frac{|\psi(s)|^2}{s}\mathrm{d}S<0 \tag{10-2}$$

即基本小波在频域也具有好的衰减性质。

一组小波基函数是通过尺度因子和位移因子由基本小波产生的，即

$$\psi_{a,b}(x)=\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{10-3}$$

连续小波变换也称为积分小波变换，定义为

$$W_f(a,b)=\left\langle f,\psi_{a,b}(x)\right\rangle=\int_{-\infty}^{\infty}f(x)\,\psi_{a,b}(x)\mathrm{d}t=\frac{1}{\sqrt{a}}\int_{-\infty}^{\infty}f(x)\,\psi_{a,b}\left(\frac{x-b}{a}\right)\mathrm{d}x \tag{10-4}$$

其逆变换为

$$f(x)=\frac{1}{C_{\psi}}\int_0^{\infty}\int_{-\infty}^{\infty}W_f(a,b)\psi_{a,b}(x)\mathrm{d}b\frac{\mathrm{d}a}{a^2} \tag{10-5}$$

二维连续小波基函数定义为

$$\psi_{ab_xb_y}(x,y)=\frac{1}{|a|}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right) \tag{10-6}$$

二维连续小波变换为

$$W_f(a,b_x,b_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\ \psi_{ab_xb_y}(x,y)\mathrm{d}x\mathrm{d}y \tag{10-7}$$

二维连续小波逆变换为

$$f(x,y)=\frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W_f(a,b_x,b_y)\ \psi_{ab_xb_y}(x,y)\ \mathrm{d}b_x\mathrm{d}b_y\frac{\mathrm{d}a}{a^3} \tag{10-8}$$

1．滤波器族

这里将小波变换与一组带通线性（卷积）滤波器相联系，来解释小波变换的基本原理。

首先定义尺度 a 上的一般小波基函数：

$$\psi_a(x)=\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right) \tag{10-9}$$

这是用 a 作尺度因子，并用 $a^{-1/2}$ 作为模范的基本小波。若记其翻转共轭为

$$\begin{aligned}\tilde{\psi}_a(x)&=\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right)\\ \tilde{\psi}_a(x)&=\psi_a^{\phi}(x)=\frac{1}{\sqrt{a}}\psi^{\phi}\left(-\frac{x}{a}\right)\end{aligned} \tag{10-10}$$

小波变换就可以表示成滤波器族：

$$W_f(a,b)\int_{-\infty}^{\infty} f(x)\tilde{\psi}_a(b-x)\mathrm{d}x=f*\tilde{\psi}_a \tag{10-11}$$

而且每个滤波器的输出分量再次滤波并适当伸缩后组合在一起可重构 $f(x)$，即

$$\begin{aligned}f(x)&=\frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty}[f*\tilde{\psi}_a](b)\psi_a(b-x)\frac{\mathrm{d}^2a}{a^2}\\ &=\frac{1}{C_\psi}\int_0^{\infty}[f*\tilde{\psi}_a*\psi_a](x)\frac{\mathrm{d}a}{a^2}\end{aligned} \tag{10-12}$$

图 10-1 表示了对一个信号的小波变换的滤波族分析。

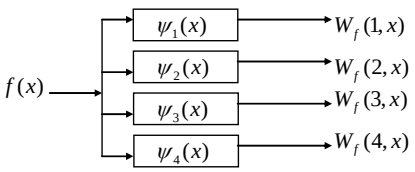


图 10-1　一个信号的小波变换滤波族分析

2．二维滤波器族

在二维情况下，每个滤波器都是一个二维冲激响应，输入是图像上的带通滤波器，滤波后的图像的叠层组成了小波变换。图 10-2 对二维滤波器族作了说明。

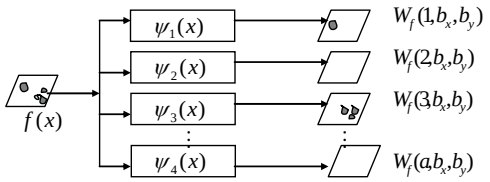


图 10-2　二维滤波器族示意图

MATLAB 实现了信号的一维连续小波变换，可以通过函数 cwt 实现相应的功能。其调用格式如下：

COEFS = cwt(S,SCALES,'wname')

COEFS = cwt(S,SCALES,'wname','plot')

COEFS = cwt(S,SCALES,'wname',PLOTMODE)

COEFS = cwt(S,SCALES,'wname',PLOTMODE,XLIM)

【例 10-1】下面以信号 noissin 为例说明如何对一个信号进行连续小波分解，信号 noissin 是一个含噪声的周期性信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load noissin;          %装载 noissin 信号
x=noissin;
figure;plot(x);        %绘制原始 noissin 信号
title('原始 noissin 信号');
figure;
% 用 db4 小波函数进行一维连续小波变换
subplot(2,1,1);c=cwt(x,1:48,'db4','plot');
% 重新选择尺度后进行一维连续小波变换
subplot(2,1,2);c=cwt(x,2:2:128,'db4','plot');
figure;
c=cwt(x,2:2:128,'cgau4','plot'); %使用复小波对其进行连续小波变换
```

运行程序，效果如图 10-3～图 10-5 所示。

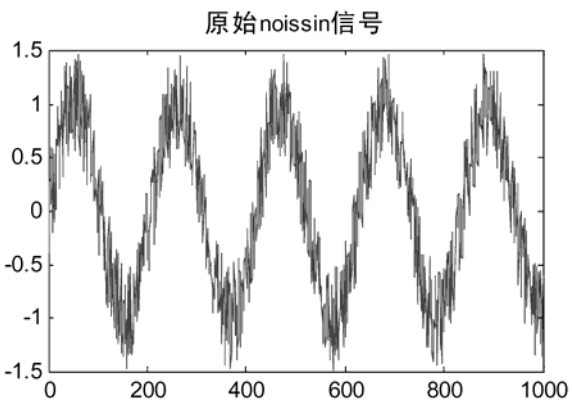


图 10-3　原始 noissin 信号

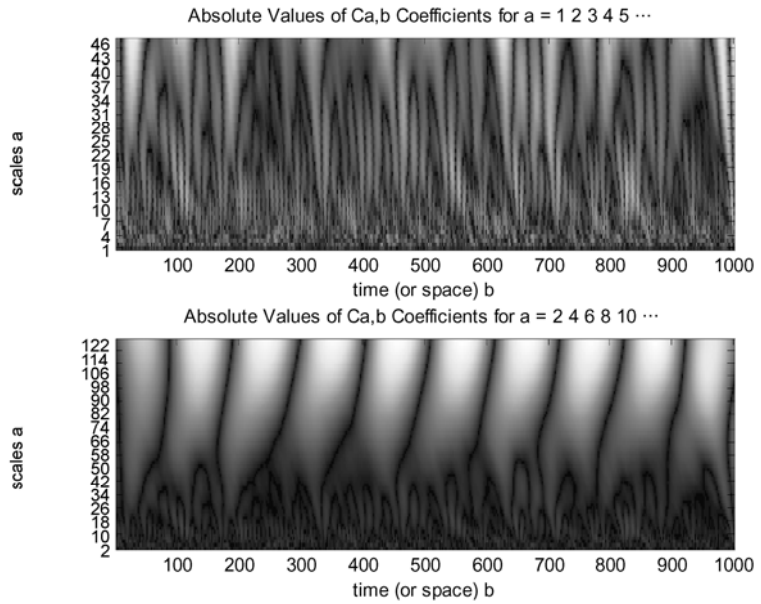


图 10-4　不同尺度下的连续小波分解

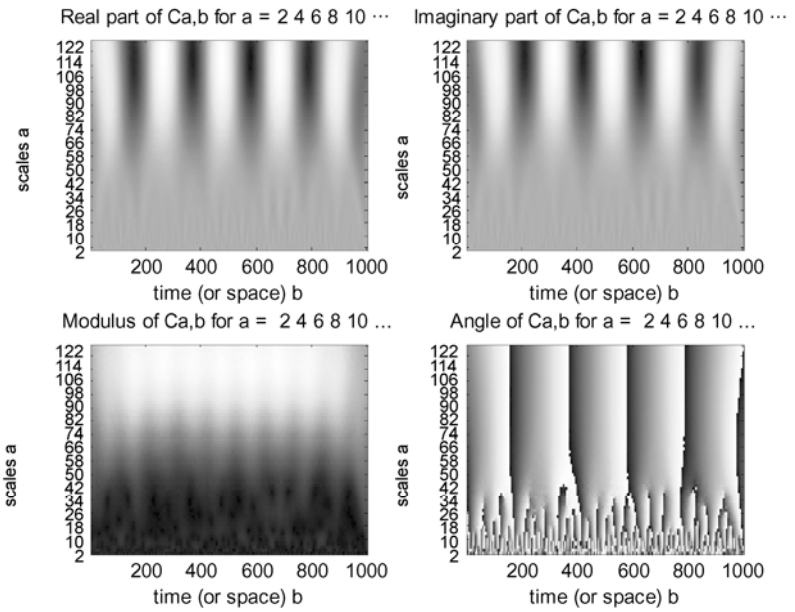


图 10-5　信号的复连续小波分解

10.1.2　信号的离散小波变换

在图像处理中，需把连续小波及其小波变换离散化才有意义。作为一种方便的形式，计算机实现中往往对离散小波进行二进制离散，并把经这种离散化后的小波及相应的小波变换称为离散小波变换（DWT）。离散小波变换是对连续小波变换的尺度和位移接照 2 的幂次进行离散化得到的，故又称为二进制小波变换。

在每个可能的尺度下计算小波系数，计算量相当大，将产生惊人的数据，于是考虑选择部

分膨胀和位移来进行计算。使用二进制膨胀和位移，就使分析十分有效，并且相当精确。

虽然傅里叶变换可以反映信号的整个内涵，但是表现形式却不够直观，而且噪声会导致信号频谱复杂。在信号处理领域一直都是使用一族带通滤波器将信号分解为不同频率分量，即将输入信号 $f(t)$并行送到通滤波器族 $H_i(x)$，则

$$g_i(x)=\int_{-\infty}^{+\infty} f(t)H_i(x-t)\mathrm{d}t \tag{10-13}$$

构造一族滤波器 $H_i(x)$时要满足如下条件：

$$\sum_{i=1}^{\infty}H_i(x)=1\Rightarrow\sum_{i=1}^{\infty}g_i(x)=f(x) \tag{10-14}$$

这里 $g_i(x)$就是一组小波变换系数，而$\{H_i(x)\}$就是一个小波函数集。

滤波器族理论提供了一个振荡信号分析的方便手段，但在图像分析中，人们关心的并不是真正的振荡信号，而是信号的一个或一部分周期。为了更仔细地观察这部分信息，可以使用不同的小波变换尺度来膨胀或收缩小波基，从而实现信号的多分辨率。

在实际运用中，尤其是在计算机上实现时，连续小波必须加以离散化。因此，有必要讨论连续小波 $\psi_{a,b}(t)$和连续小波变换 $W_f(a,b)$的离散化。需要强调指出的是，这一离散化都是针对连续的尺度参数 a 和连续平移参数 b 的，而不是针对时间变量 t 的。这一点与以前习惯的时间离散化不同。在连续小波中，考虑函数

$$\psi_{a,b}(t)=|a|^{-1/2}\psi\left(\frac{t-b}{a}\right)$$

这里 $b\in R$ ， $a\in R^{+}$，且 $a\neq 0$， ψ是容性的，为方便起见，在离散化中，总限制 a 只取正值，这样相容性条件就变为

$$C_{\psi}=\int_0^{\infty}\frac{|\hat{\psi}(\bar{\omega})|}{|\bar{\omega}|}\mathrm{d}\bar{\omega}<\infty \tag{10-15}$$

通常，把连续小波变换中尺度参数 a 和平移参数 b 的离散公式分别取做 $a=a_0^j, b=ka_0^jb_0$，这里 $j\in Z$，扩展步长 $a_0\neq 1$是固定值。为方便起见，总是假定 $a_0>1$（由于 m 可取正也可取负，所以这个假定无关紧要），所以对应的离散小波函数 $\psi_{j,k}(t)$ 即可写为

$$\psi_{j,k}(t)=a_0^{-j/2}\psi\left(\frac{t-ka_0^jb_0}{a_0^j}\right)=a_0^{-j/2}\psi(a_0^{-j}t-kb_0) \tag{10-16}$$

而离散化小波变换系数则可表示为

$$C_{j,k}=\int_{-\infty}^{\infty}f(t)\psi_{j,k}^{*}(t)\mathrm{d}t=<f,\psi_{j,k}> \tag{10-17}$$

其重构公式为

$$f(t)=C\sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty}C_{j,k}\psi_{j,k}(t) \tag{10-18}$$

C 是一个与信号无关的常数。然而，怎样选择 a_0 和 b_0，才能够保证重构信号的精度呢？显然，网格点应尽可能密（即 a_0 和 b_0 尽可能小），因为如果网格点越稀疏，使用的小波函数 $\psi_{j,k}(t)$ 和离散小波系数 $C_{j,k}$ 就越少，信号重构的精度也就会越低。

MATLAB 实现了信号的单尺度一维离散小波变换、信号的多尺度一维离散小波变换及提取

信号的多尺度离散小波分解的高频和低频系数。

1．dwt 函数

功能：实现单尺度离散小波变换的函数。其调用格式如下：

[ca, cd]=dwt(X, 'wname')

[ca, cd]=dwt(X, Lo_D, Hi_D)

[ca, cd]=dwt(X, 'wname', 'mode', MODE)

[ca, cd]=dwt(X, Lo_D, Hi_D, 'mode', MODE)

2．wavedec 函数

功能：实现多尺度一维离散小波变换函数。其调用格式如下：

[C,L] = wavedec(X,N,'wname')

[C,L] = wavedec(X,N,Lo_D,Hi_D)

3．appcoef 函数

功能：提取多尺度小波变换的低频系数。其调用格式如下：

A = appcoef(C,L,'wname',N)

A = appcoef(C,L,'wname')

A = appcoef(C,L,Lo_R,Hi_R)

A = appcoef(C,L,Lo_R,Hi_R,N)

4．detcoef 函数

功能：提取多尺度小波变换的高频系数。其调用格式如下：

D = detcoef(C,L,N)

D = detcoef(C,L)

【例 10-2】多尺度分解并提取小波分解的低频系数和高频系数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load leleccum;       %装载 leleccum 信号
s=leleccum(1:3920);
% 用 db1 小波函数对信号进行三尺度小波分解
[C,L]=wavedec(s,2,'db1');
figure; plot(s);
title('原始 leleccum 信号');
% 提取尺度 1 的低频系数
ca1=appcoef(C,L,'db1',1);
% 提取尺度 2 的低频系数
ca2=appcoef(C,L,'db1',2);
figure;
subplot(2,1,1);plot(ca1);
title('尺度 1 的低频系数');
subplot(2,1,2);plot(ca2);
title('尺度 2 的低频系数');
% 提取尺度 1 的高频系数
cd1=detcoef(C,L,1);
% 提取尺度 2 的高频系数
cd2=detcoef(C,L,2);
```

```
figure;
subplot(2,1,1);plot(cd1);
title('尺度 1 的高频系数');
subplot(2,1,2);plot(cd2);
title('尺度 2 的高频系数');
```

运行程序，效果如图 10-6～图 10-8 所示。

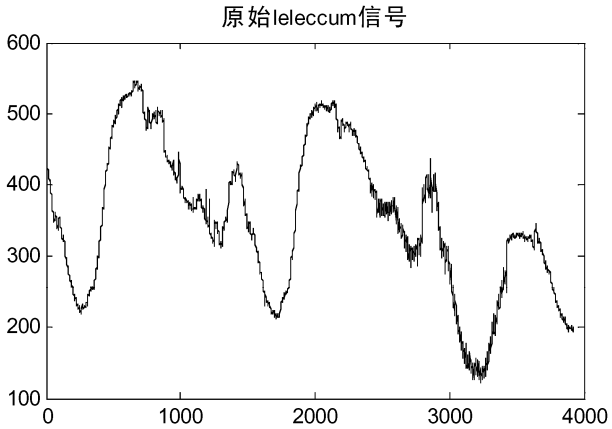


图 10-6　原始 leleccum 信号

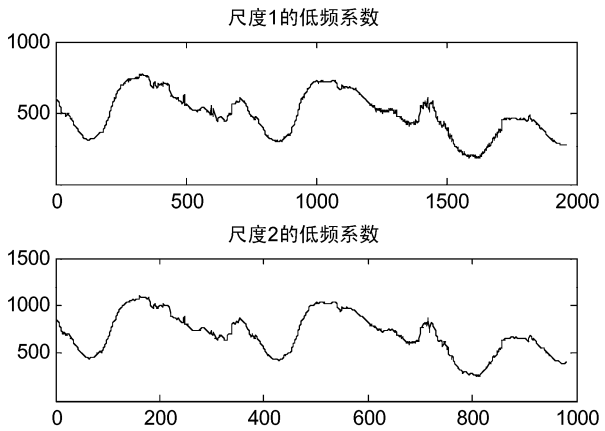


图 10-7　尺度 1 和尺度 2 的低频系数

实际计算中不可能对全部尺度因子值和位移参数值计算 CWT 的 a,b 值，加之实际的观测信号都是离散的，所以信号处理中都是用离散小波变换（DWT）。大多数情况下是将尺度因子和位移参数按 2 的幂次进行离散。最有效的计算方法是 s. Mallat 于 1988 年发展的快小波算法（又称塔式算法）。对任一信号，离散小波变换第一步运算是将信号分为低频部分（称为近似部分）和高频部分（称为细节部分）。近似部分代表了信号的主要特征。第二步对低频部分再进行相似运算，不过这时尺度因子已经改变，依次进行到所需要的尺度。除了连续小波（CWT）、离散小波（DWT），还有小波包（Wavelet Packet）和多维小波。

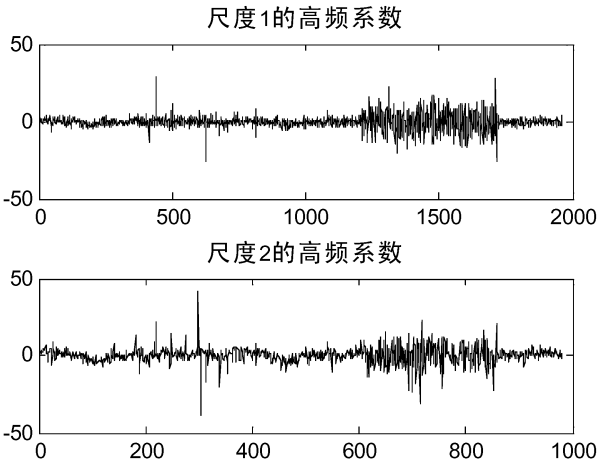


图 10-8　尺度 1 和尺度 2 的高频系数

【例 10-3】通过对 vonkoch 的操作，体现离散和连续小波变换的区别。

其实现的 MATLAB 程序代码如下：

```
>> load vonkoch
vonkoch=vonkoch(1:510);
lv = length(vonkoch);
subplot(311), plot(vonkoch);
title('Analyzed signal.');
set(gca,'Xlim',[0 510]);
[c,l] = wavedec(vonkoch,5,'sym2');
cfd = zeros(5,lv);
for k = 1:5
      d = detcoef(c,l,k);
      d = d(ones(1,2^k),:);
      cfd(k,:) = wkeep(d(:)',lv);
end
cfd = cfd(:);
I = find(abs(cfd)<sqrt(eps));
cfd(I)=zeros(size(I));
cfd = reshape(cfd,5,lv);
% 绘制离散系数
subplot(312), colormap(pink(64));
img = image(flipud(wcodemat(cfd,64,'row')));
set(get(img,'parent'),'YtickLabel',[]);
title('Discrete Transform, absolute coefficients.')
ylabel('level')
subplot(313)
ccfs = cwt(vonkoch,1:32,'sym2','plot');
title('Continuous Transform, absolute coefficients.')
colormap(pink(64));
ylabel('Scale')
```

运行程序，效果如图 10-9 所示。

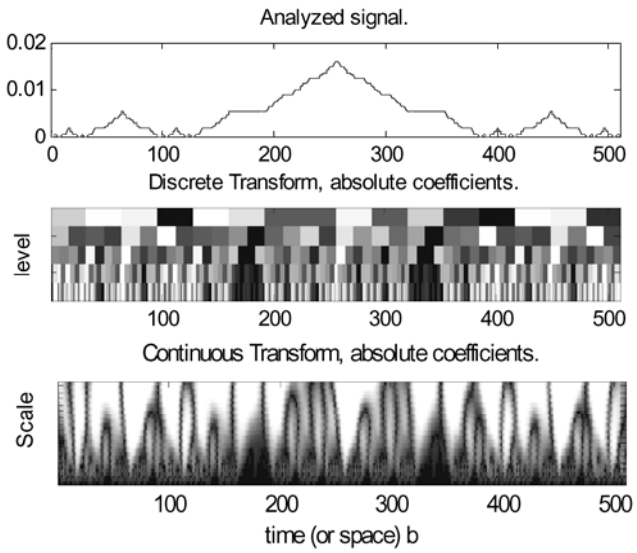


图 10-9　连续小波与离散小波比较效果

10.1.3　信号的小波包

在多分辨分析中，$L^2(R)=\oplus_{j\in z}W_j$，表明多分辨分析是按照不同的尺度因子 j 把 Hilbert 空间 $L^2(R)$ 分解为所有子空间 $W_j(j\in Z)$ 的正交和的，其中，W_j 为小波函数 $\psi(t)$ 的闭包（小波子空间）。现在，对小波子空间 W_j 按照二进制分式进行频率的细分，以达到提高频率分辨率的目的。

一种自然的做法是将尺度空间 V_j 和小波子空间 W_j 用一个新的子空间 U_j^n 统一起来表征，若令

$$\begin{cases} U_j^0=V_j \\ U_j^1=W_j \end{cases} \qquad j\in Z$$

则 Hilbert 空间的正交分解 $V_{j+1}=V_j\oplus W_j$ 即可用 U_j^n 的分解统一为

$$U_{j+1}^0=U_j^0\oplus U_j^1 \qquad j\in Z \tag{10-19}$$

定义子空间 U_j^n 是函数 $U_n(t)$ 的闭包空间，而 $U_n(t)$ 是函数 $U_{2n}(t)$ 的闭包空间，并令 $U_n(t)$ 满足下面的双尺度方程：

$$\begin{cases} u_{2n}(t)=\sqrt{2}\sum_{k\in Z}h(k)u_n(2t-k) \\ u_{2n+1}(t)=\sqrt{2}\sum_{k\in Z}g(k)u_n(2t-k) \end{cases} \tag{10-20}$$

式中，$g(k)=(-1)^k h(1-k)$，即两系数也具有正交关系。当 n=0 时，式（10-20）直接给出

$$\begin{cases} u_0(t)=\sum_{k\in Z}h_k u_0(2t-k) \\ u_1(t)=\sum_{k\in Z}g_k u_0(2t-k) \end{cases} \tag{10-21}$$

与在多分辨分析中，$\phi(t)$和 $\psi(t)$ 满足双尺度方程

$$\begin{cases}\phi(t)=\sum_{k\in Z}h_k\phi(2t-k) & \{h_k\}_{k\in Z}\in l^2\\ \psi(t)=\sum_{k\in Z}g_k\phi(2t-k) & \{g_k\}_{k\in Z}\in l^2\end{cases} \tag{10-22}$$

相比较，$u_0(t)$ 和 $u_1(t)$ 分别退化为尺度函数 $\phi(t)$ 和小波基函数 $\psi(t)$。式（10-21）是式（10-19）的等价表示。把这种等价表示推广到 $n\in Z_+$（非负整数）的情况，即得到式（10-20）的等价表示为

$$U_{j+1}^n=U_j^n\oplus U_j^{2n+1}\qquad j\in Z\ ;\ \ n\in Z_+ \tag{10-23}$$

定义（小波包） 由式（10-20）构造的序列 $\{u_n(t)\}$（其中 $n\in Z_+$）称为由基函数 $u_0(t)=\phi(t)$ 确定的正交小波包。当 n=0 时，即为式（10-21）的情况。

由于 $\phi(t)$ 由 h_k 唯一确定，所以又称 $\{u_n(t)\}_{n\in Z}$ 为关于序列 $\{h_k\}$ 的正交小波包。

MATLAB 实现了信号的一维小波包分解和提取一维小波包分解的系数。

1．wpdec 函数

功能：实现一维小波包的分解。其调用格式如下：

T = wpdec(X,N,'wname',E,P)

T = wpdec(X,N,'wname')

2．wpcoef 函数

功能：实现提取小波包分解系数。其调用格式如下：

X = wpcoef(T,N)

X = wpcoef(T)

【例 10-4】下面以信号 noisdopp 为例说明如何进行信号的小波包分解。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load noisdopp;      % 装载 signal 信号
x = noisdopp;
figure(1); subplot(211);
plot(x); title('原始信号 signal');
% 使用 db1 小波包对信号 x 进行 3 层分解
% 使用 Shannon 熵
wpt = wpdec(x,3,'db1');
% 绘制小波分解树结构
plot(wpt)
% 计算结点(2,1)的系数
cfs = wpcoef(wpt,[2 1]);
figure(1); subplot(212);
plot(cfs); title('结点(2,1)系数');
```

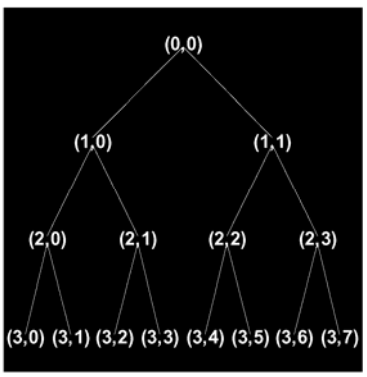


图 10-10 小波分解树结构

小波分解树结构如图 10-10 所示。运行程序，效果如图 10-11 所示。

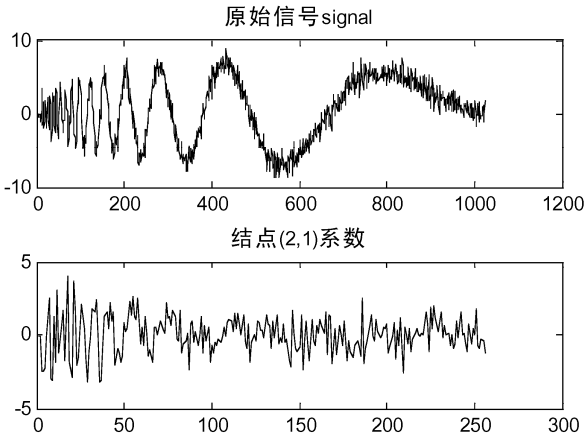


图 10-11　结点（2，1）的系数

10.2　信号重构

10.2.1　信号的小波重构

MATLAB 实现了一维离散小波分解重构，主要包括单尺度一维离散小波变换的逆变换、多尺度小波变换的重构、对一维小波分解结构进行低频或高频重构、一维小波系数的直接重构以及一维小波分解的单尺度重构。

1．idwt 函数

功能：实现单尺度一维离散小波变换的逆变换。其调用格式如下：

X = idwt(cA,cD,'wname')

X = idwt(cA,cD,Lo_R,Hi_R)

X = idwt(cA,cD,'wname',L)

X = idwt(cA,cD,Lo_R,Hi_R,L)

X = idwt(...,'mode',MODE)

【例 10-5】单尺度一维小波分解的重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
randn('seed',531316785);              %定义随机信号的状态
s = 2 + kron(ones(1,8),[1 -1]) + ...
      ((1:16).^2)/32 + 0.2*randn(1,16);
% 用小波函数 db2 对信号进行单尺度一维小波分解
[ca1,cd1] = dwt(s,'db2');
subplot(221); plot(ca1);
title('利用 db2 重构低通');
subplot(222); plot(cd1);
title('利用 db2 重构高通');
ss = idwt(ca1,cd1,'db2');
err = norm(s-ss);          % Check reconstruction
subplot(212); plot([s;ss]');
```

```
title('原始信号及重构后的信号的误差');
xlabel(['重构误差 ',num2str(err)])
[Lo_R,Hi_R] = wfilters('db2','r');
ss = idwt(ca1,cd1,Lo_R,Hi_R);
```

运行程序，效果如图 10-12 所示。

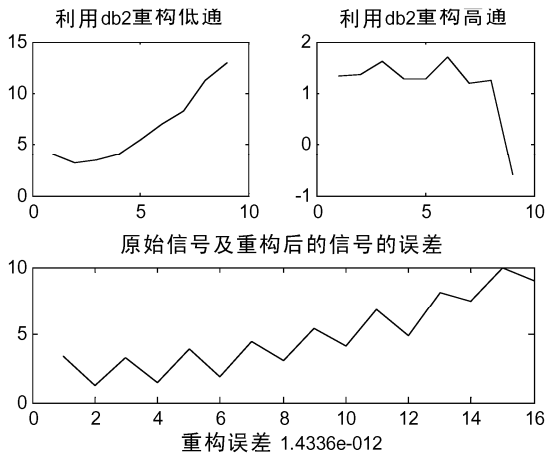


图 10-12　单尺度一维分解结构

2．waverec 函数

功能：多尺度小波变换的重构。其调用格式如下：

X = waverec(C,L,'wname')

X = waverec(C,L,Lo_R,Hi_R)

3．wrcoef 函数

功能：小波分解结构的低频或高频重构。其调用格式如下：

X = wrcoef('type',C,L,'wname',N)

X = wrcoef('type',C,L,Lo_R,Hi_R,N)

X = wrcoef('type',C,L,'wname')

X = wrcoef('type',C,L,Lo_R,Hi_R)

【例 10-6】信号的多尺度小波分解重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load leleccum;        %装载原始 leleccum 信号
s=leleccum(1:3920);
% 用小波函数 db1 对信号进行三尺度小波分解
[C,L]=wavedec(s,3,'db1');
subplot(2,1,1);plot(s);
title('原始 leleccum 信号');
 % 用小波函数 db1 进行信号的低频重构
 a3=wrcoef('a',C,L,'db1');
 subplot(2,1,2);plot(a3);
 title('小波重构信号');
```

运行程序，效果如图 10-13 所示。

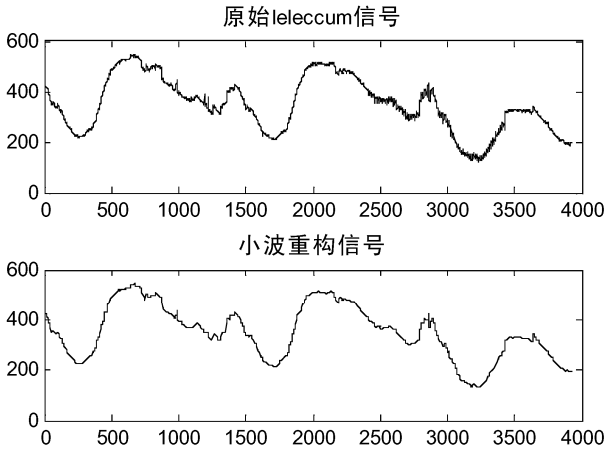


图 10-13　信号的小波分解低频重构

4．upcoef 函数

功能：小波系数的直接重构。其调用格式如下：

Y = upcoef(O,X,'wname',N)

Y = upcoef(O,X,'wname',N,L)

Y = upcoef(O,X,Lo_R,Hi_R,N)

Y = upcoef(O,X,Lo_R,Hi_R,N,L)

Y = upcoef(O,X,'wname')

Y = upcoef(O,X,Lo_R,Hi_R)

【例 10-7】通过小波分解系数进行小波重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
cfs = [1];
essup = 10;
figure(1)
for i=1:6
    rec = upcoef('a',cfs,'db6',i);
    ax = subplot(6,1,i),h = plot(rec(1:essup));
    set(ax,'xlim',[1 325]);
    essup = essup*2;
end
subplot(611)
title(['单尺度低频系数向上 1 to 6 重构信号']);
cfs = [1];
mi = 12; ma = 30;
rec = upcoef('d',cfs,'db6',1);
figure(2)
subplot(611), plot(rec(3:12))
for i=2:6
    rec = upcoef('d',cfs,'db6',i);
    subplot(6,1,i), plot(rec(mi*2^(i-2):ma*2^(i-2)))
end
```

```
subplot(611)
title(['单尺度高频系数向上 1 to 6 重构信号']);
```

运行程序，效果如图 10-14 及图 10-15 所示。

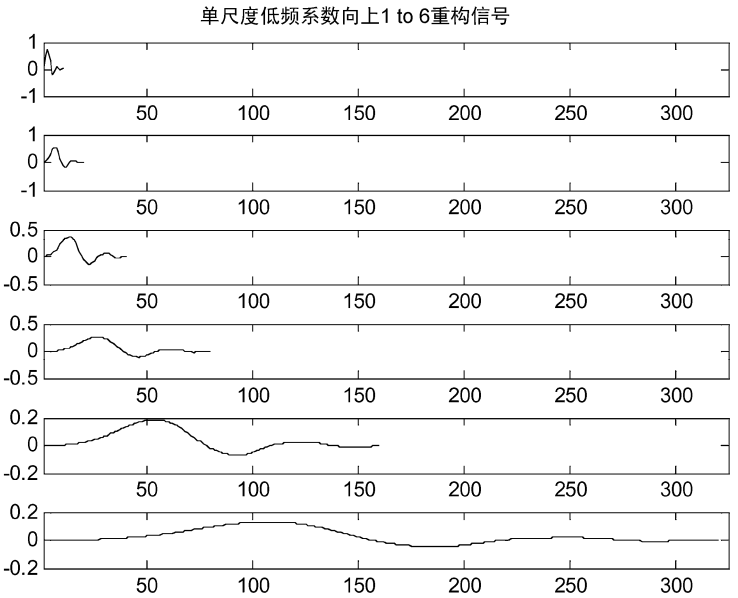


图 10-14　低频系数的重构

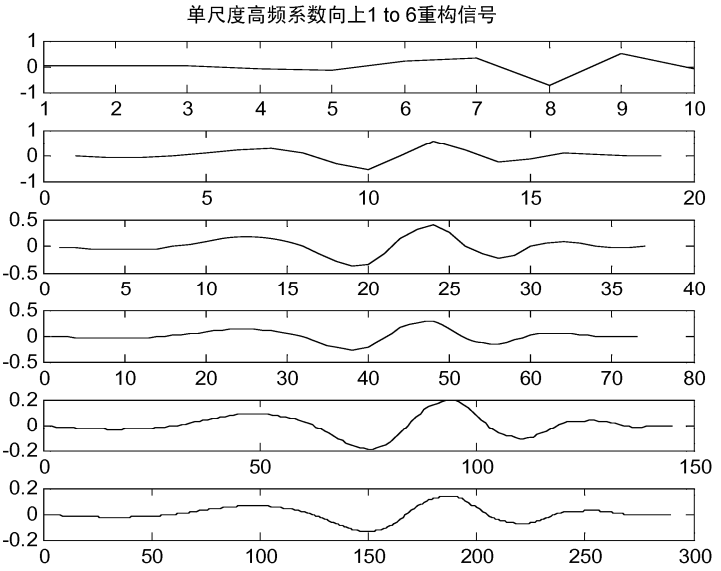


图 10-15　高频系数的重构

5．upwlev 函数

功能：实现一维小波分解的单尺度重构。其调用格式如下：

[NC,NL,cA] = upwlev(C,L,'wname')

[NC,NL,cA] = upwlev(C,L,Lo_R,Hi_R)

【例 10-8】对小波分解进行单尺度重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load sumsin;            % 装载原始 sumsin 信号
s = sumsin;
% 用小波函数 db1 对信号进行三尺度小波分解
[c,l] = wavedec(s,3,'db1');
subplot(311); plot(s);
title('原始 sumsin 信号');
subplot(312); plot(c);
title('小波 3 层重构')
xlabel(['尺度 3 的低频系数和尺度 3,2,1 的高频系数'])
% 获得尺度 2 的小波分解
[nc,nl] = upwlev(c,l,'db1');
subplot(313); plot(nc);
title('小波 2 层重构')
xlabel(['尺度 2 的低频系数和尺度 2,1 的高频系数'])
```

运行程序，效果如图 10-16 所示。

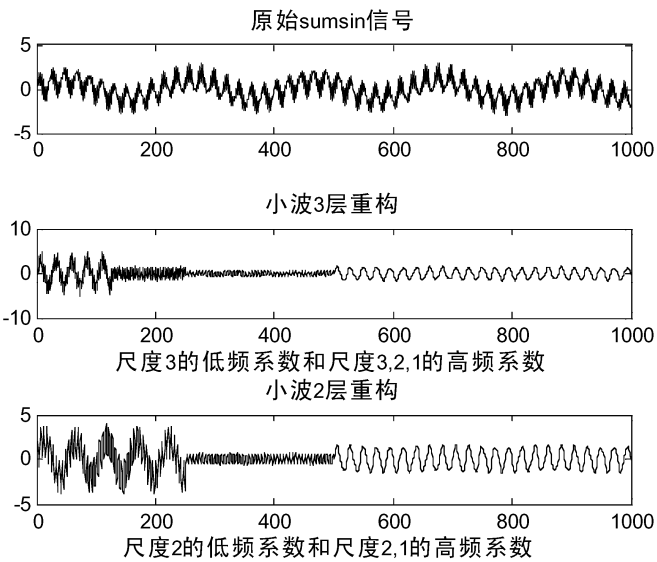


图 10-16　小波分解的单尺度重构

10.2.2　信号的小波包重构

MATLAB 实现了一维小波分解的重构及对小波包某一结点系数进行重构，分别介绍如下。

1．wprec 函数

功能：实现一维小波包分解的重构。其调用格式如下：

X = wprec(T)

【例 10-9】对小波包分解的树结构进行重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load sumsin;            % 装载原始 sumsin 信号
s = sumsin;
% 用小波函数 db2 对信号分解互第三层
% 使用 Shannon 熵
wpt=wpdec(s,3,'db2','shannon');
% 对信号进行重构
rex=wprec(wpt);
subplot(211); plot(s);
title('原始 sumsin 信号');
subplot(212); plot(rex);
title('重构后的信号');
```

运行程序，效果如图 10-17 所示。

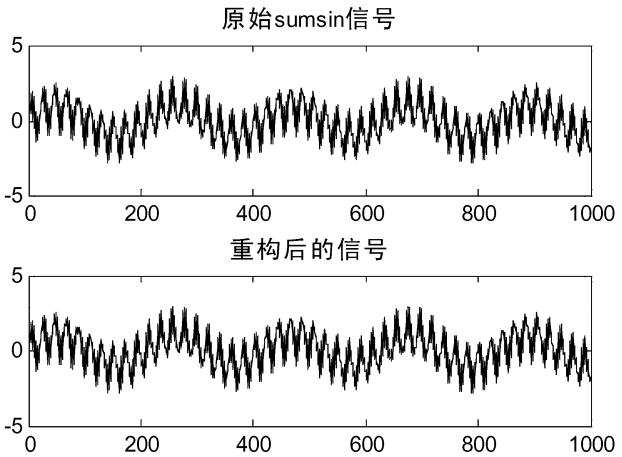


图 10-17　小波包分解重构

2．wprcoef 函数

功能：实现小波包某一结点系数的重构。其调用格式如下：

X = wprcoef(T,N)

【例 10-10】对小波包分解的某一结点系数进行重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load noisdopp;     %装载原始 noisdopp 信号
x = noisdopp;
% 使用 db1 小波包对信号 x 进行 3 层分解
t = wpdec(x,3,'db1','shannon');
subplot(2,1,1);plot(x)
title('原始 noisdopp 信号');
% 重构小波包结点(2,1)
rcfs = wprcoef(t,[2 1]);
subplot(212); plot(rcfs);
title('重构小波包结点(2,1)');
```

运行程序，效果如图 10-18 所示。

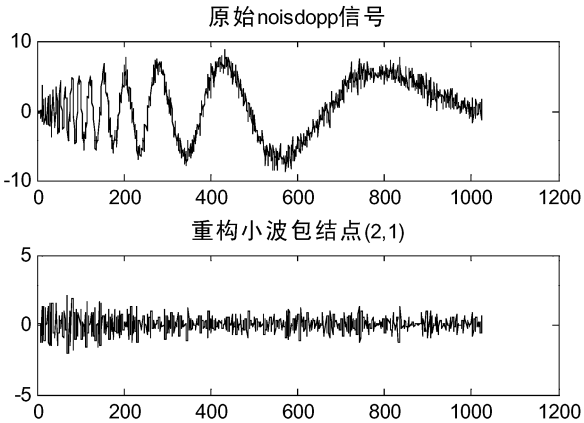


图 10-18　小波包结点重构

10.3　信号分析

10.3.1　分离信号的不同成分

在本节中将通过两个例子说明小波分析在离散信号的不同成分中的应用。

1．正弦信号加噪声

【例 10-11】下面通过使用小波分析一个由正弦信号（正弦信号的周期约为 200）加白噪声组成的信号，说明小波分析如何分离这两种信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load noissin;      %装载原始 noissin 信号
s=noissin;
figure;
subplot(6,1,1);plot(s);
ylabel('s');
% 使用 db5 小波对信号进行 5 层分解
[C,L]=wavedec(s,5,'db5');
for i=1:5
    % 对分解的第 5 层到第 1 层的低频系数进行重构
    a=wrcoef('a',C,L,'db5',6-i);
    subplot(6,1,i+1); plot(a);
    ylabel(['a',num2str(6-i)]);
end
figure;
subplot(6,1,1);plot(s);
ylabel('s');
for i=1:5
    % 对分解的第 5 层到第 1 层的高频系数进行重构
    d=wrcoef('d',C,L,'db5',6-i);
    subplot(6,1,i+1);plot(d);
```

```
    ylabel(['d',num2str(6-i)]);
end
```

运行程序，效果如图 10-19 及图 10-20 所示。

如图 10-20 所示，小波分解的细节信号是由白噪声分解得到的，而正弦信号可以在图 10-19 的近似信号 a5 中得到。因为在这一层噪声对信号的影响已经可以忽略了。

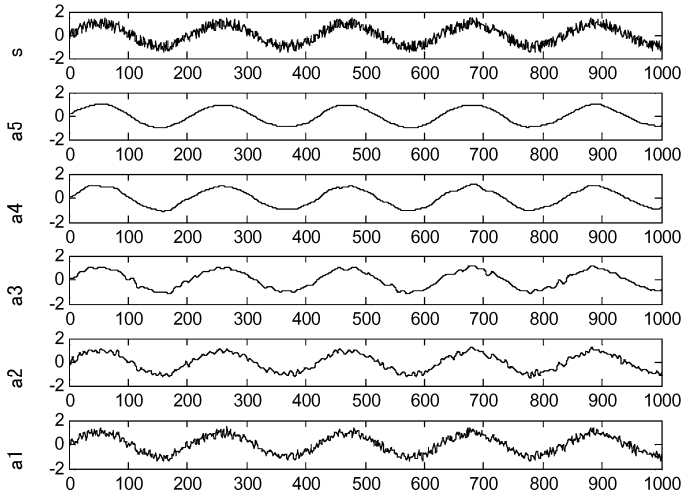


图 10-19　小波分解的低频系数

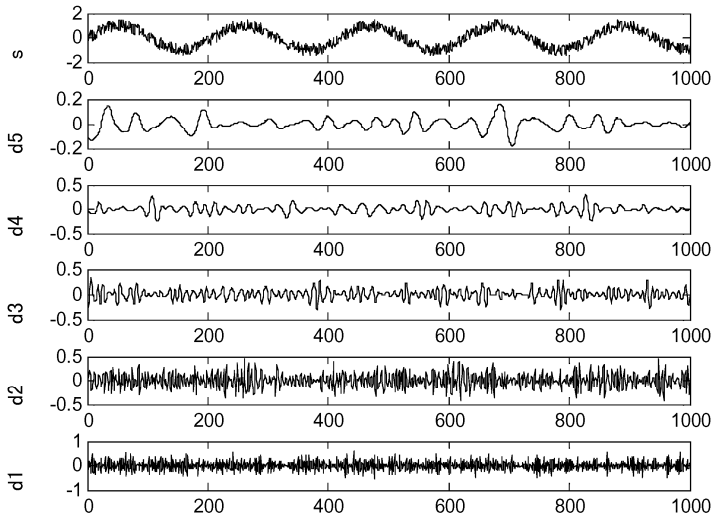


图 10-20　小波分解的高频系数

2．正弦信号加三角波

【例 10-12】下面通过使用小波分析一个由正弦信号（正弦信号的周期约为 20）加三角波组成的信号，说明小波分析如何分离这两种信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load trsin;      %装载原始 trsin 信号
s=trsin;
figure;
```

```
subplot(7,1,1);plot(s);
ylabel('s');axis tight;
% 使用 db5 小波对信号进行 6 层分解
[C,L]=wavedec(s,6,'db5');
for i=1:6
    % 对分解的第 6 层到第 1 层的低频系数进行重构
    a=wrcoef('a',C,L,'db5',7-i);
    subplot(7,1,i+1); plot(a);
    axis tight;
    ylabel(['a',num2str(7-i)]);
end
figure;
subplot(7,1,1);plot(s);
ylabel('s'); axis tight;
for i=1:6
    % 对分解的第 6 层到第 1 层的高频系数进行重构
    d=wrcoef('d',C,L,'db5',7-i);
    subplot(7,1,i+1);plot(d);
    ylabel(['d',num2str(7-i)]);
    axis tight;
end
```

运行程序，效果如图 10-21 及图 10-22 所示。

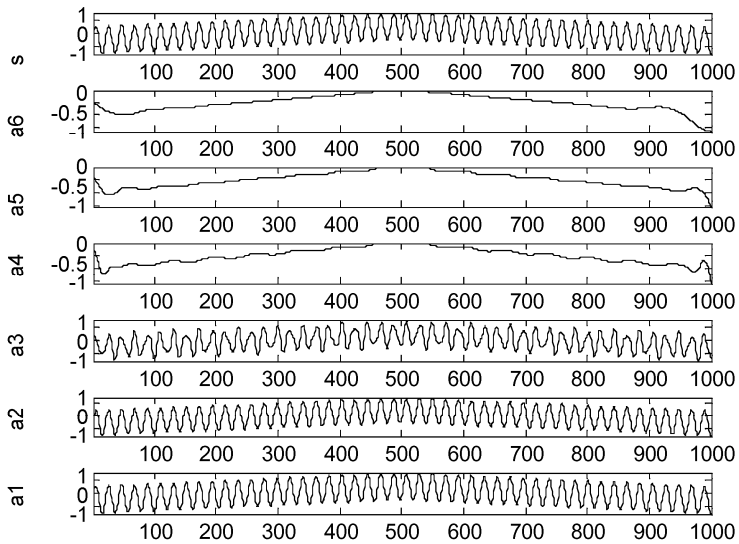


图 10-21　小波分解的低频系数

如图 10-21 和图 10-22 所示，d1、d2、d5、d6 的值非常小，这说明信号中包含的这些频率段的信号非常少，d3 和 d4 主要是由正弦信号产生的，a6 主要是由三角波分解产生的。

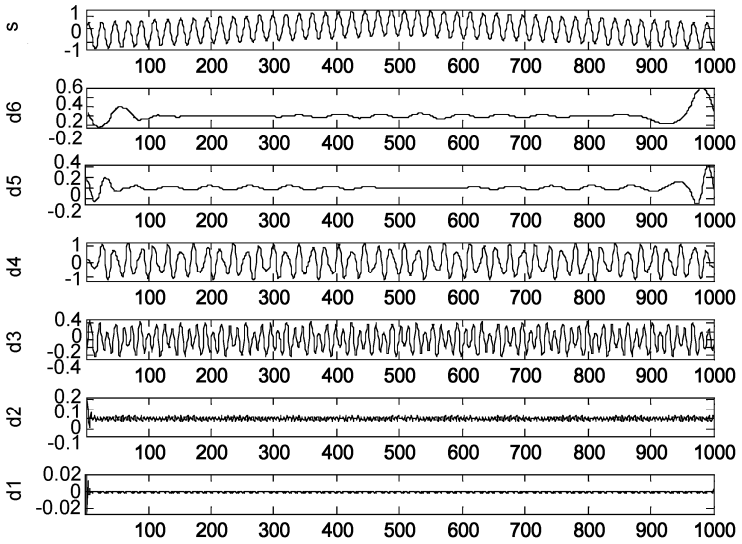


图 10-22　小波分解的高频系数

10.3.2　识别某一频率区间内的信号

下面通过一个例子说明如何应用小波分析识别某一频率区间内的信号。

【例 10-13】使用小波来分析一个由三个不同频率的正弦信号叠加的信号，看是否能够将这三个正弦信号区分开来，结果证明小波分析可以很好地识别某一频率区间内的信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load sumsin;      %装载原始 sumsin 信号
s=sumsin;
figure;
subplot(6,1,1);plot(s);
ylabel('s');title('原始信号及各层近似信号');
%  使用 db3 小波对信号进行 5 层分解
[C,L]=wavedec(s,5,'db3');
for i=1:5
    % 对分解的第 5 层到第 1 层的低频系数进行重构
    a=wrcoef('a',C,L,'db3',6-i);
    subplot(6,1,i+1); plot(a);
    ylabel(['a',num2str(6-i)]);
end
figure;
subplot(6,1,1);plot(s);
ylabel('s'); title('原始信号及各层细节信号');
for i=1:5
    % 对分解的第 6 层到第 1 层的高频系数进行重构
    d=wrcoef('d',C,L,'db3',6-i);
    subplot(6,1,i+1);plot(d);
    ylabel(['d',num2str(6-i)]);
end
```

运行程序，效果如图 10-23 及图 10-24 所示。

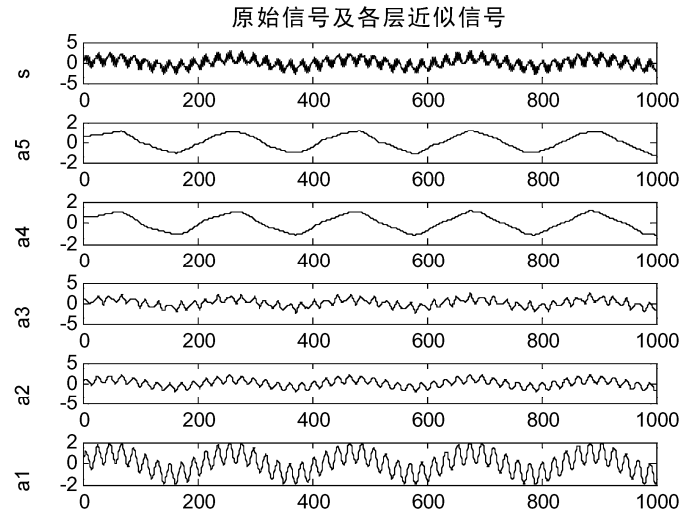


图 10-23　小波分解近似信号

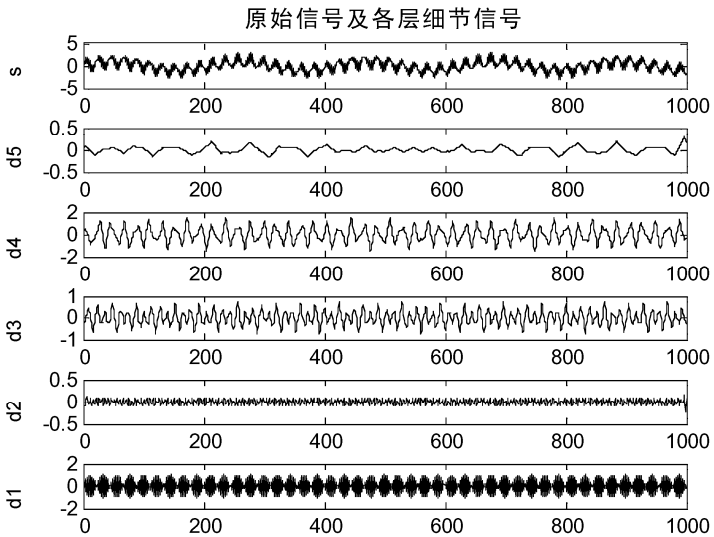


图 10-24　小波分解的细节信号

本例中，该信号是由周期分别为 200、20 和 2 的三个正弦信号合成，它们的采样周期均为 1，为方便叙述，在此分别称为低频、中频和高频正弦信号。可以看出，低频、中频和高频信号分别对应于分解的近似信号 a4、细节信号 d4 以及细节信号 d1。

细节信号 d1 包含的主要是周期在 1 和 2 之间的信号，但这直接从图 10-24 上无法看出。将 d1 放大，如图 10-25 所示，可以看到每个椭圆形中包含 10 次振动，依此来估计周期，可以发现它接近于 2。

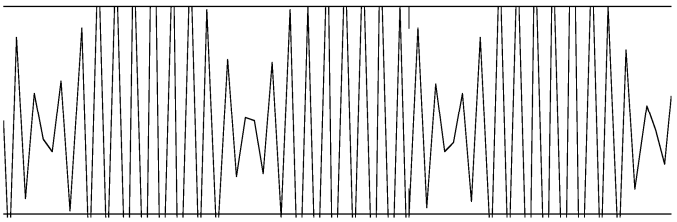


图 10-25　细节信号 d1

细节信号 d3 和细节信号 d4 包含中频信号。注意到在近似信号 a3 和 a4 之间出现了信号的不连续，这是因为中频信号是由这两层共同表达的。中频信号的周期的估计需要使用近似信号 a1 和 a3。将近似 a1 放大，如图 10-26 所示，可以看到周期为 20 的中频信号。而周期为 2 的低频信号在图 10-23 的近似信号 a4 中可以清晰地分辨出。

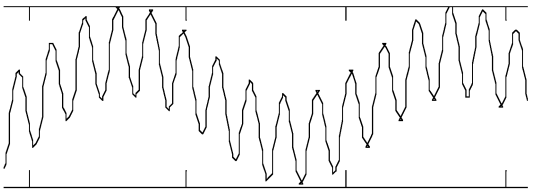


图 10-26　近似信号 a1

10.3.3　识别信号的发展趋势

有时某信号由于受到噪声污染，导致无法直接辨别信号的发展趋势。由于信号的发展趋势往往代表的是信号的低频部分，因此通过信号的多尺度分解，在分解的低频系数中可以观察到信号的发展趋势。

【例 10-14】下面对一个被有色噪声污染的谐波信号进行小波分解，说明小波分析在识别信号的发展趋势中的应用。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load cnoislop;                %装载原始 cnoislop 信号
x=cnoislop;
subplot(6,1,1);plot(x);
ylabel('x');
% 进行一维离散小波变换
[C,L]=wavedec(x,5,'db4');
for i=1:5
    % 对分解结构[C,L]中的低频部分进行重构
    s=wrcoef('a',C,L,'db4',6-i);
    subplot(6,1,i+1);plot(s);
    ylabel(['a',num2str(6-i)]);
end
```

运行程序，效果如图 10-27 所示。

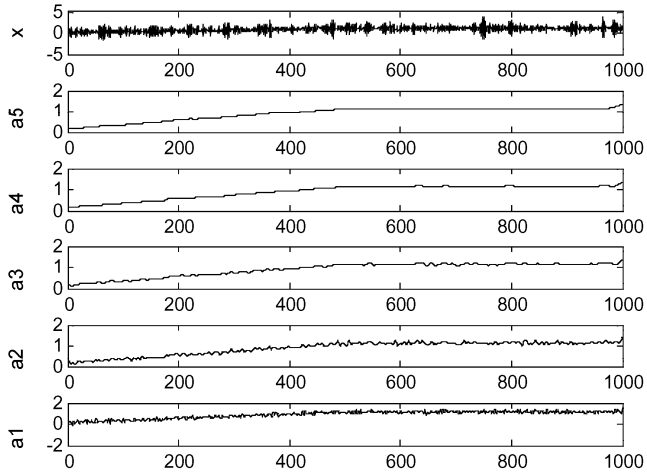


图 10-27　识别信号的发展趋势

10.4　信号去噪

10.4.1　信号阈值去噪

信号去噪是信号处理领域的经典问题之一。传统的去噪方法主要包括线性滤波方法和非线性滤波方法，如中值滤波和 Wiener 滤波等。传统去噪方法的不足在于使信号变换后的熵增高、无法刻画信号的非平稳特性并且无法得到信号的相关性。为了克服上述缺点，人们开始使用小波变换解决信号去噪问题。

小波变换具有下列良好特性：

（1）低熵性：小波系数的稀疏分布，使信号变换后的熵降低。

（2）多分辨率特性：可以非常好地刻画信号的非平稳特性，如边缘、尖峰、断点等。

（3）去相关性：可取出信号的相关性，且噪声在小波变换后有白化趋势，所以比时域更利于去噪。

（4）选基灵活性：由于小波变换可以灵活选择基函数，因此可根据信号特点和去噪要求选择适合小波。

小波在信号去领域已得到越来越广泛的应用。阈值去噪方法是一种实现简单、效果较好的小波去噪方法。阈值去噪方法的思想就是对小波分解后的各层系数中模大于和小于某阈值的系数分别处理，然后对处理完的小波系数再进行反变换，重构出经过去噪后的信号。

1．阈值获取

MATLAB 中实现信号阈值获取的函数有 ddencmp、thselect、wbmpen 和 wdcbm，下面对它们进行简单的介绍。

1）ddencmp 函数

功能：用于获取在消噪或压缩过程中的默认阈值。其调用格式如下：

[THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,IN2,X)

[THR,SORH,KEEPAPP] = ddencmp(IN1,'wv',X)

[THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,'wp',X)

【例 10-15】ddencmp 函数用法示例。

```
>> clear all;
init = 2055415866; randn('seed',init);
x = randn(1,1000);
[thr,sorh,keepapp] = ddencmp('den','wv',x)
thr =       3.8593
sorh =s
keepapp =         1
>> [thr,sorh,keepapp] = ddencmp('cmp','wv',x)
thr =       0.7003
sorh =h
keepapp =         1
>> [thr,sorh,keepapp,crit] = ddencmp('den','wp',x)
thr =       4.2911
sorh =h
keepapp =         1
crit =sure
>> [thr,sorh,keepapp,crit] = ddencmp('cmp','wp',x)
thr =       0.7003
sorh =h
keepapp =         1
crit =threshold
```

2）thselect 函数

功能：自适应阈值的获取。其调用格式如下：

THR = thselect(X,TPTR)

【例 10-16】thselect 函数用法示例。

```
>> clear all;
init = 2055415866; randn('seed',init);
x = randn(1,1000);
thr = thselect(x,'rigrsure')
thr =       1.8065
>> thr = thselect(x,'sqtwolog')
thr =       3.7169
>> thr = thselect(x,'heursure')
thr =       3.7169
>> thr = thselect(x,'minimaxi')
thr =       2.2163
```

3）wbmpen 函数

功能：用于全局阈值。其调用格式如下：

THR = wbmpen(C,L,SIGMA,ALPHA)

【例 10-17】wbmpen 函数用法示例。

```
>> clear all;
load noisbump; x = noisbump;
wname = 'sym6'; lev = 5;
[c,l] = wavedec(x,lev,wname);
sigma = wnoisest(c,l,1);
```

```
alpha = 2;
thr = wbmpen(c,l,sigma,alpha)
thr =
      2.7681
```

4）wdcbm 函数

功能：用于使用 Birge-Massart 算法获取一维小波变换的阈值。其调用格式如下：

[THR,NKEEP] = wdcbm(C,L,ALPHA)

[THR,NKEEP] = wdcbm(C,L,ALPHA,M)

【例 10-18】wdcbm 函数用法示例。

```
>> clear all;
load leleccum;
indx = 2600:3100;
x = leleccum(indx);
wname = 'db3'; lev = 5;
[c,l] = wavedec(x,lev,wname);
alpha = 1.5; m = l(1);
[thr,nkeep] = wdcbm(c,l,alpha,m)
thr =
    19.5569    17.1415    20.2599    42.8959    15.0049
nkeep =
      1      2      3      4      7
```

2．信号的阈值去噪

MATLAB 中实现信号的阈值去噪的函数有 wden、wdencmp、wthresh、wthcoef、wpthcoef 及 wpdencmp。下面对它们进行简单的介绍。

1）wden 函数

功能：用于一维信号的自动消噪。其调用格式如下：

[XD,CXD,LXD] = wden(X,TPTR,SORH,SCAL,N,'wname')

[XD,CXD,LXD] = wden(C,L,TPTR,SORH,SCAL,N,'wname')

2）wdencmp 函数

功能：用于一维或二维信号的消噪或压缩。其调用格式如下：

[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('gbl',X,'wname',N,THR,SORH,KEEPAPP)

[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',X,'wname',N,THR,SORH)

[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',C,L,'wname',N,THR,SORH)

3）wthresh 函数

功能：用于软硬阈值的处理。其调用格式如下：

Y = wthresh(X,SORH,T)

【例 10-19】wthresh 函数用法示例。

```
>> clear all;
y = linspace(-1,1,100);
thr = 0.4;
ythard = wthresh(y,'h',thr);
ytsoft = wthresh(y,'s',thr);
subplot(1,3,1);plot(y);
```

```
title('原始信号');
subplot(1,3,2);plot(ythard);
title('硬阈值处理');
subplot(1,3,3);plot(ytsoft);
title('软阈值处理');
```

运行程序，效果如图 10-28 所示。

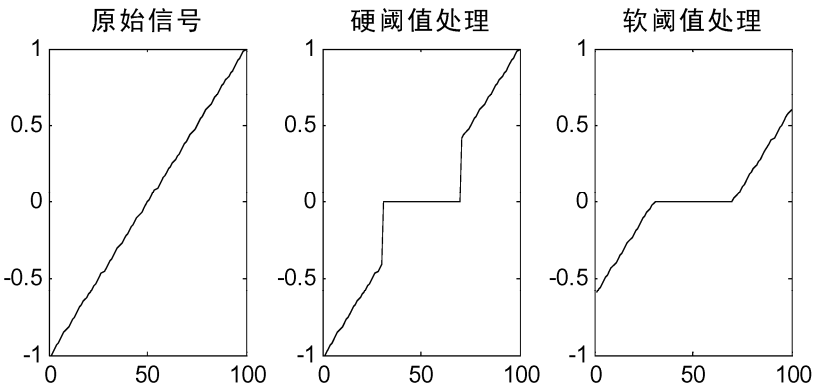


图 10-28　wthresh 函数处理效果

4）wthcoef 函数

功能：对小波包树进行阈值处理。其调用格式如下：

NC = wthcoef('d',C,L,N,P)

NC = wthcoef('d',C,L,N)

NC = wthcoef('a',C,L)

NC = wthcoef('t',C,L,N,T,SORH)

5）wpthcoef 函数

功能：用于一维信号小波系数的阈值处理。其调用格式如下：

T = wpthcoef(T,KEEPAPP,SORH,THR)

6）wpdencmp 函数

功能：用于使用小波包变换进行信号的压缩或去噪。其调用格式如下：

[XD,TREED,PERF0,PERFL2] = wpdencmp(X,SORH,N,'wname',CRIT,PAR,KEEPAPP)

[XD,TREED,PERF0,PERFL2] =wpdencmp(TREE,SORH,CRIT,PAR,KEEPAPP)

10.4.2　信号阈值去噪应用

一般说来，信号去噪的基本步骤主要包括如下三步：

（1）信号的小波分解。

（2）小波分解高频系数的阈值量化。

（3）信号的小波重构。使用小波分解的低频系数以及阈值量化处理后的高频系数进行小波重构。

【例 10-20】使用函数 ddencmp 获取信号去噪阈值，然后采用函数 wdencmp 实现信号去噪。其实现的 MATLAB 程序代码如下：

```
>> clear all;
load sinsin
```

```
init=2055615866;  %产生含噪声信号
randn('seed',init);
x = X + 18*randn(size(X));
%获取消噪的阈值
[thr,sorh,keepapp] = ddencmp('den','wv',x);
%  对信号进行消噪处理
xd = wdencmp('gbl',x,'sym4',2,thr,sorh,keepapp);
subplot(2,2,1); plot(X);
title('原始信号');
subplot(2,2,2);plot(x);
title('含噪声信号');
subplot(2,2,3);plot(xd);
ylim([-0 100]);
title('消噪后的信号');
```

运行程序，效果如图 10-29 所示。

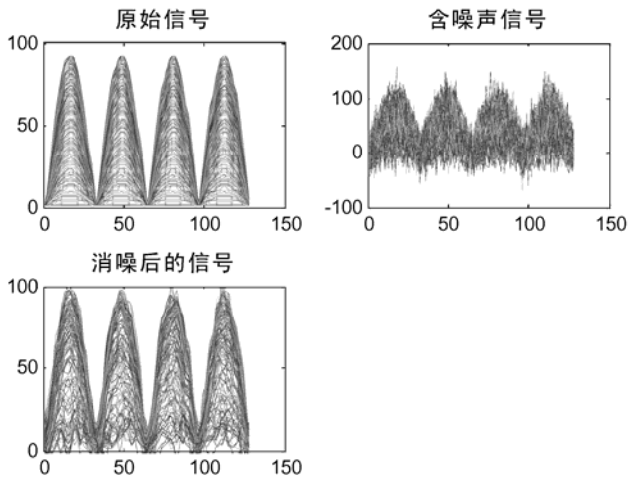


图 10-29　一维信号去噪效果

【例 10-21】对小波分解系数使用函数 wthcoef 进行阈值处理，然后利用阈值处理后的小波系数进行小波重构达到去噪的目的。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load leleccum;
indx=1:1024;
x=leleccum(indx);
%产生含噪声信号
init=2055615866;
randn('seed',init);
nx = x + 18*randn(size(x));
%  将信号 nx 用小波函数 sym5 分解到第 5 层
%  用 minimaxi 阈值选择对系数进行处理,消除噪声信号
lev=5;
xd=wden(nx,'minimaxi','s','mln',lev,'sym5');
subplot(2,2,1); plot(x);
```

```
title('原始信号');
subplot(2,2,2);plot(nx);
title('含噪声信号');
subplot(2,2,3);plot(xd);
title('消噪后的信号');
```

运行程序，效果如图 10-30 所示。

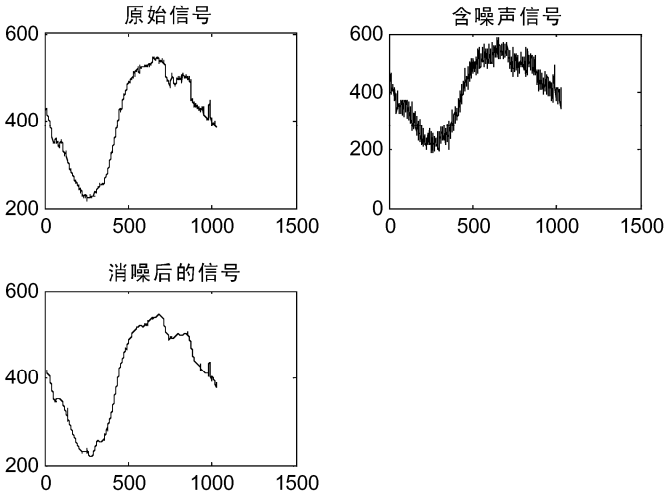


图 10-30　一维信号的自动消噪

10.5　提升小波变换用于信号处理

提升方案是 Sweldens 和 Daubechiesa 等学者于 20 世纪 90 年代中期提出的。基于提升方案的小波变换称为提升小波变换。

10.5.1　提升小波变换概述

在提升小波变换中，小波不一定是由某一母小波通过膨胀和平移得到的，它们的定义非常灵活，也许是在某一区间，也许是定义在不规则的网格上。在二维空间中，提升小波变换更加灵活、复杂，不是简单、规律地划分二维平面，而可能是定义在某一曲面上，如定义在球面上的小波。提升方案的另外一个优点在于它能够包容传统小波，也就是说，所有的传统小波都可以用提升方案构造出来。通过 Eucliden 算法，所有的传统小波可以由提升方案中基本的提升和偶分解而成。采用提升方案实现小波滤波具有以下优点：

（1）运算速度快。

（2）不需要额外的内存。

（3）可以实现整数小波变换。

提升小波变换由于其计算速度快，占用内存少，可以实现整数变换等特点，因而被作为 JPEG2000 里面的核心算法。其基本过程可以分为分裂、预测和更新三个环节。通过预测和更新两个提升环节实现信号的高低频分离。由于信号有局部相关性，某一点的信号值可以通过其相邻信号的值以适当的预测算子预测出来，同时预测出来的误差就是高频的信息，从而这个过程就是预测环节。预测环节下面得到的高频信息又通过更新算子来调整信号的下抽样来得到低

频信息，这个过程就是更新环节。

10.5.2　提升小波

1．提升小波变换

MATLAB 中实现提升小波变换的函数是 lwt 和 lwt2。其中 lwt 实现了一维提升小波变换，lwt2 实现了二维提升小波变换。

1）lwt 函数

功能：实现一维提升小波变换。其调用格式如下：

```
[CA,CD] = lwt(X,W)
X_InPlace = lwt(X,W)
lwt(X,W,LEVEL)
X_InPlace = lwt(X,W,LEVEL,'typeDEC',typeDEC)
[CA,CD] = lwt(X,W,LEVEL,'typeDEC',typeDEC)
```

【例 10-22】lwt 函数用法示例。

```
>> clear all;
lshaar = liftwave('haar');
els = {'p',[-0.125 0.125],0};
lsnew = addlift(lshaar,els);
x = 1:8;
[cA,cD] = lwt(x,lsnew)
lshaarInt = liftwave('haar','int2int');
lsnewInt = addlift(lshaarInt,els);
[cAint,cDint] = lwt(x,lsnewInt)
```

运行程序，输出如下：

```
cA =
    1.9445    4.9497    7.7782   10.6066
cD =
    0.7071    0.7071    0.7071    0.7071
cAint =
     1     3     5     7
cDint =
     1     1     1     1
```

2）lwt2 函数

功能：实现二维提升小波变换。其调用格式如下：

```
[CA,CH,CV,CD] = lwt2(X,W)
X_InPlace = lwt2(X,LS)
lwt2(X,W,LEVEL)
X_InPlace = lwt2(X,W,LEVEL,'typeDEC',typeDEC)
[CA,CD] = lwt2(X,W,LEVEL,'typeDEC',typeDEC)
```

【例 10-23】lwt2 函数用法示例。

```
>> clear all;
lshaar = liftwave('haar');
els = {'p',[-0.125 0.125],0};
```

```
lsnew = addlift(lshaar,els);
x = reshape(1:16,4,4);
[cA,cH,cV,cD] = lwt2(x,lsnew)
lshaarInt = liftwave('haar','int2int');
lsnewInt = addlift(lshaarInt,els);
[cAint,cHint,cVint,cDint] = lwt2(x,lsnewInt)
```

运行程序，输出如下：

```
cA =
    5.7500   22.7500
   10.0000   27.0000
cH =
    1.0000    1.0000
    1.0000    1.0000
cV =
    4.0000    4.0000
    4.0000    4.0000
cD =
     0     0
     0     0
cAint =
     3    11
     5    13
cHint =
     1     1
     1     1
cVint =
     4     4
     4     4
cDint =
     0     0
     0     0
```

2．提升小波的逆变换

MATLAB 中实现提升小波逆变换的函数是 ilwt 和 ilwt2。其中 ilwt 实现了一维提升小波逆变换，ilwt2 实现二维提升小波变换。

1）ilwt 函数

功能：实现一维提升小波的逆变换。其调用格式如下：

```
X = ilwt(AD_In_Place,W)
X = ilwt(CA,CD,W)
X = ilwt(AD_In_Place,W,LEVEL)
X = ilwt(CA,CD,W,LEVEL)
X = ilwt(AD_In_Place,W,LEVEL,'typeDEC',typeDEC)
X = ilwt(CA,CD,W,LEVEL,'typeDEC',typeDEC)
```

【例 10-24】ilwt 函数用法示例。

```
>> clear all;
lshaar = liftwave('haar');
```

```
els = {'p',[-0.125 0.125],0};
lsnew = addlift(lshaar,els);
x = 1:8;
[cA,cD] = lwt(x,lsnew);
lshaarInt = liftwave('haar','int2int');
lsnewInt = addlift(lshaarInt,els);
[cAint,cDint] = lwt(x,lsnewInt);
xRec = ilwt(cA,cD,lsnew);
err = max(max(abs(x-xRec)))
xRecInt = ilwt(cAint,cDint,lsnewInt);
errInt = max(max(abs(x-xRecInt)))
```

运行程序，输出如下：

```
err =    4.4409e-016
errInt =         0
```

2）ilwt2 函数

功能：为二维提升小波逆变换函数。其调用格式如下：

X = ilwt2(AD_In_Place,W)

X = ilwt2(CA,CH,CV,CD,W)

X = ilwt2(AD_In_Place,W,LEVEL)

X = ilwt2(CA,CH,CV,CD,W,LEVEL)

X = ilwt2(AD_In_Place,W,LEVEL,'typeDEC',typeDEC)

X = ilwt2(CA,CH,CV,CD,W,LEVEL,'typeDEC',typeDEC)

【例 10-25】ilwt2 函数用法示例。

```
>> clear all;
lshaar = liftwave('haar');
els = {'p',[-0.125 0.125],0};
lsnew = addlift(lshaar,els);
x = reshape(1:16,4,4);
[cA,cH,cV,cD] = lwt2(x,lsnew);
lshaarInt = liftwave('haar','int2int');
lsnewInt = addlift(lshaarInt,els);
[cAint,cHint,cVint,cDint] = lwt2(x,lsnewInt);
xRec = ilwt2(cA,cH,cV,cD,lsnew);
err = max(max(abs(x-xRec)))
xRecInt = ilwt2(cAint,cHint,cVint,cDint,lsnewInt);
errInt = max(max(abs(x-xRecInt)))
```

运行程序，输出如下：

```
err =         0
errInt =         0
```

3．提升小波变换系数的提取或重构

MATLAB 中实现提升小波变换系数的提升或重构的函数是 lwtcoef 和 lwtcoef2。

1）lwtcoef 函数

功能：实现一维提升小波变换系数的提取或重构。其调用格式如下：

Y = lwtcoef(TYPE,XDEC,LS,LEVEL,LEVEXT)

Y = lwtcoef(TYPE,XDEC,W,LEVEL,LEVEXT)

【例 10-26】lwtcoef 函数用法示例。

```
>> clear all;
lshaar = liftwave('haar');
els = {'p',[-0.125 0.125],0};
lsnew = addlift(lshaar,els);
x = 1:8;
xDec = lwt(x,lsnew,2)
xDec =
 4.3438    0.7071    2.1250    0.7071   13.0313    0.7071    2.0000    0.7071
>> ca1 = lwtcoef('ca',xDec,lsnew,2,1)
ca1 =
    1.9445    4.9497    7.7782   10.6066
>> a1 = lwtcoef('a',xDec,lsnew,2,1)
a1 =
  1.3750    1.3750    3.5000    3.5000    5.5000    5.5000    7.5000    7.5000
>> a2 = lwtcoef('a',xDec,lsnew,2,2)
a2 =
 2.1719    2.1719    2.1719    2.1719    6.5156    6.5156    6.5156    6.5156
>> d1 = lwtcoef('d',xDec,lsnew,2,1)
d1 =
  -0.3750    0.6250   -0.5000    0.5000   -0.5000    0.5000   -0.5000    0.5000
>> d2 = lwtcoef('d',xDec,lsnew,2,2)
d2 =
 -0.7969   -0.7969    1.3281    1.3281   -1.0156   -1.0156    0.9844    0.9844
>> err = max(abs(x-a2-d2-d1))
err =   9.9920e-016
```

2）lwtcoef2 函数

功能：lwtcoef2 实现二维小波变换系数的提取或重构。其调用格式如下：

Y = lwtcoef2(TYPE,XDEC,LS,LEVEL,LEVEXT)

Y = lwtcoef2(TYPE,XDEC,W,LEVEL,LEVEXT)

【例 10-27】lwtcoef2 函数示例。

```
>> clear all;
lshaar = liftwave('haar');
els = {'p',[-0.125 0.125],0};
lsnew = addlift(lshaar,els);
x = reshape(1:16,4,4);
xDec = lwt2(x,lsnew,2)
xDec =
   27.4375    4.0000   17.0000    4.0000
    1.0000         0    1.0000         0
    4.2500    4.0000    0.0000    4.0000
    1.0000         0    1.0000         0
ca1 = lwtcoef2('ca',xDec,lsnew,2,1)
ca1 =
    5.7500   22.7500
```

```
    10.0000    27.0000
>> a2 = lwtcoef2('a',xDec,lsnew,2,2)
a2 =
     6.8594    6.8594    6.8594    6.8594
     6.8594    6.8594    6.8594    6.8594
     6.8594    6.8594    6.8594    6.8594
     6.8594    6.8594    6.8594    6.8594
>> h1 = lwtcoef2('h',xDec,lsnew,2,1)
h1 =
    -0.3750   -0.3750   -0.3750   -0.3750
     0.6250    0.6250    0.6250    0.6250
    -0.5000   -0.5000   -0.5000   -0.5000
     0.5000    0.5000    0.5000    0.5000
>> v1 = lwtcoef2('v',xDec,lsnew,2,1)
v1 =
    -1.5000    2.5000   -2.0000    2.0000
    -1.5000    2.5000   -2.0000    2.0000
    -1.5000    2.5000   -2.0000    2.0000
    -1.5000    2.5000   -2.0000    2.0000
>> d1 = lwtcoef2('d',xDec,lsnew,2,1)
d1 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> h2 = lwtcoef2('h',xDec,lsnew,2,2)
h2 =
    -0.7969   -0.7969   -0.7969   -0.7969
    -0.7969   -0.7969   -0.7969   -0.7969
     1.3281    1.3281    1.3281    1.3281
     1.3281    1.3281    1.3281    1.3281
>> v2 = lwtcoef2('v',xDec,lsnew,2,2)
v2 =
    -3.1875   -3.1875    5.3125    5.3125
    -3.1875   -3.1875    5.3125    5.3125
    -3.1875   -3.1875    5.3125    5.3125
    -3.1875   -3.1875    5.3125    5.3125
>> d2 = lwtcoef2('d',xDec,lsnew,2,2)
d2 =
   1.0e-015 *
     0.2498    0.2498   -0.4163   -0.4163
     0.2498    0.2498   -0.4163   -0.4163
    -0.4163   -0.4163    0.6939    0.6939
    -0.4163   -0.4163    0.6939    0.6939
>> err = max(max(abs(x-a2-h2-v2-d2-h1-v1-d1)))
err =
   3.5527e-015
```

10.5.3 提升小波在信号处理中的应用

1．信号分解

与传统的小波分解相比，提升小波可以实现整数小波变换，这对于许多应用而言是非常重要的性质。

【例 10-28】提取小波分解的某一层小波系数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
lsharr=liftwave('haar');          %得到 Haar 小波的提升方案
els={'p',[-0.125 0.125],0};       %添加 els 到提升方案中
lsnew=addlift(lsharr,els);
% 2 层提升小波分解
load noisdopp;
x=noisdopp;
xdec=lwt(x,lsnew,2);
ca1=lwtcoef('ca',xdec,lsnew,2,1);     %提取第 1 层的近似系数
ca2=lwtcoef('ca',xdec,lsnew,2,2);     %提取第 2 层的近似系数
cd1=lwtcoef('cd',xdec,lsnew,2,1);     %提取第 1 层的细节系数
cd2=lwtcoef('cd',xdec,lsnew,2,2);     %提取第 2 层的细节系数
subplot(3,1,1);plot(x);
title('原始 noisdopp 信号');
subplot(3,2,3);plot(ca1);
title('第一层近似信号');
subplot(3,2,4);plot(ca2);
title('第二层近似信号');
subplot(3,2,5);plot(cd1);
title('第一层细节信号');
subplot(3,2,6);plot(cd2);
title('第二层细节信号');
```

运行程序，效果如图 10-31 所示。

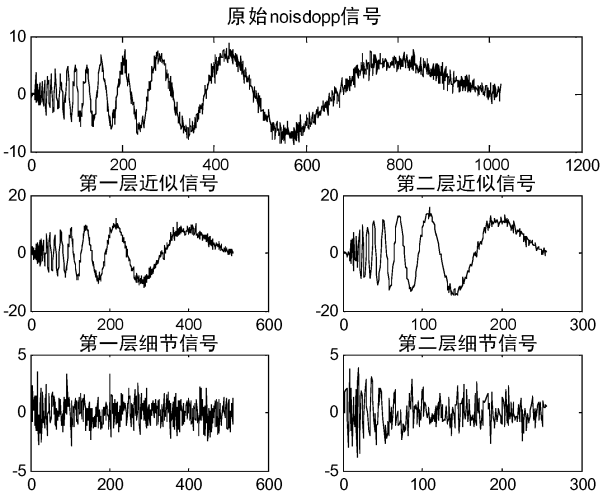


图 10-31　提取提升分解系数

2．信号重构

【例 10-29】对提升小波分解的某一层系数进行单支重构。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
lsharr=liftwave('haar');            %得到 Haar 小波的提升方案
els={'p',[-0.125 0.125],0};         %添加 els 到提升方案中
lsnew=addlift(lsharr,els);
% 2 层提升小波分解
load noisdopp;
x=noisdopp;
xdec=lwt(x,lsnew,2);
a1=lwtcoef('a',xdec,lsnew,2,1);    %提取第 1 层的近似系数
a2=lwtcoef('a',xdec,lsnew,2,2);    %提取第 2 层的近似系数
d1=lwtcoef('d',xdec,lsnew,2,1);    %提取第 1 层的细节系数
d2=lwtcoef('d',xdec,lsnew,2,2);    %提取第 2 层的细节系数
%  检查重构误差
err=max(abs(x-a2-d2-d1))
subplot(3,1,1);plot(x);
title('原始 noisdopp 信号');
subplot(3,2,3);plot(a1);
title('重构第一层近似信号');
subplot(3,2,4);plot(a2);
title('重构第二层近似信号');
subplot(3,2,5);plot(d1);
title('重构第一层细节信号');
subplot(3,2,6);plot(d2);
title('重构第二层细节信号');
```

运行程序，输出如下，效果如图 10-32 所示。

```
err =
   1.9984e-015
```

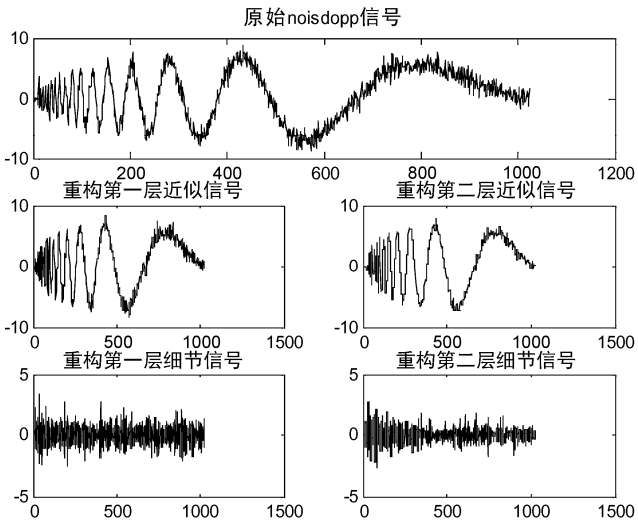


图 10-32　提升小波分解单支重构

第 11 章　MATLAB 在数字信号中的应用

11.1　雷达信号的产生

现代雷达的体制多种多样，根据雷达体制不同，可选用各种各样的信号形式。雷达信号形式的不同，对发射机的射频部分和调制器的要求也不同。对于常规雷达的简单脉冲波形而言，调制器主要要满足脉冲宽度、脉冲重复频率和脉冲波形（脉冲上升边、下降边和顶部的不稳定）的要求，一般困难不大，但是对于复杂调制、射频放大器和调制器往往要采用一些特殊的措施才能满足要求。

11.1.1　脉冲幅度调制

脉冲幅度调制或者脉冲调制（PAM）是指脉冲序列的幅度随信息信号线性变化的一种调制方式。类比于正弦波的调幅信号，脉冲幅度调制（PAM）信号可以表示为

$$s_{\text{pam}}(t) = [A_0 + f(t)]s_p(t) \qquad (11\text{-}1)$$

式中：A_0 为常数，代表直流电平；$f(t)$ 为信息信号，通常为正弦波；$s(t)$ 为脉冲序列，其波形可以是任意的，但在一般分析和实际应用中，多采用矩形波，且单极性更为广泛。实现脉冲幅度调制的方法比较简单，一般将信号 $[A_0 + f(t)]$ 与矩形脉冲序列 $s_p(t)$ 直接相乘而得到的信号就是脉冲幅度调制（PAM）信号。

因为根据脉冲幅度调制的定义，可以写出产生脉冲幅度调制信号产生的源代码如下：

```
function sp=pam(t,fc,fp,fs,tao,pha)
%该程序是用来产生脉冲调制信号(PAM)的
%参数 t 是所要产生脉冲幅度调制信号的时间,单位为 s
%参数 Fc 为脉内信号的频率,单位为 Hz
%参数 Fp 为脉冲信号的频率,单位为 Hz
%参数 Fs 为采样时钟频率,单位为 Hz
%参数 tao 为脉冲信号的占空比,单位为%
%参数 pha 为信号的初始相位,单位为 rad
if nargin<=6;   %如果不输入 tao 和 pha
    tao=50;
    pha=0;
end;
n=0:1/fs:1/fp;
tn=0:1/fs:t;
m=t/(1/fp);
spt=(square(2*pi*fp*n)+1)/2;
st=cos(2*pi*fc*n);
sq1=st.*spt;
sp=repmat(sq1,1,m);
```

【例 11-1】如果要产生以下参数要求的波形，可以调用该函数。

输入参数：脉冲调制信号时间长度：t=100μs。

脉冲内正弦信号频率：fc=1MHz。

脉冲重复频率：fp=100kHz。

采样频率：fs=10MHz。

脉冲占空比：τ=50%。

初始相位：θ=π/3。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0.000100;
fc=1000000;
fp=100000;
fs=10000000;
pha=pi/3;
ta0=50;
sp=pam(t,fc,fp,fs,ta0,pha);
plot(sp);
```

运行程序，效果如图 11-1 所示。

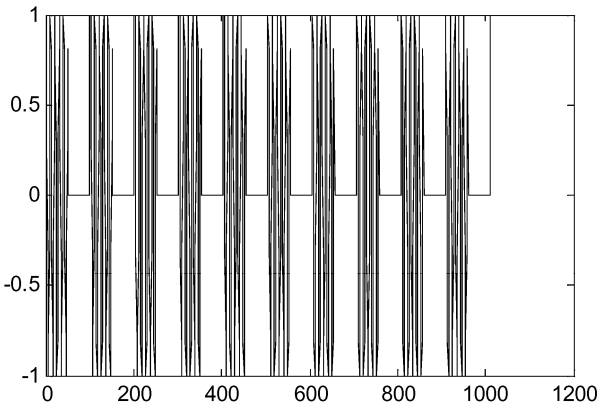


图 11-1　脉冲调制信号效果

11.1.2　线性调频信号

频率调制是指载波信号的瞬时频率偏移调制信号 $f(t)$ 线性变化的调制，即

$$\omega(t) = \omega_0 + K_{FM} f(t) \qquad (11\text{-}2)$$

式中：K_{FM} 为调频器的灵敏度。

线性调频就是载波随时间线性变化的调制信号。

在 MATLAB 工具箱中提供了 modulate 函数，可以方便地产生线性调频信号。其调用格式如下：

```
y = modulate(x,fc,fs,'method')
y = modulate(x,fc,fs,'method',opt)
[y,t] = modulate(x,fc,fs)
```

参数 x 为调制信号序列，fc 为载波频率，fs 为采样频率，method 参数是用来决定进行何种调制，method 的选择形式如下：

（1）amdsb_sc 或者 am：抑制载波双边带幅度调制。

（2）amssb：载波传输单边带幅度调制。

（3）fm：调频。

（4）pm：调相。

（5）Ppm：脉内调制。

（6）pwm：脉宽调制。

（7）qam：正交幅度调制。

（8）amssb-tc：载波传输双边带幅度调制。

【例 11-2】产生一个线性调频信号的起始频率为 10MHz，调频脉宽为 2MHz，采样频率为 10MHz，脉宽为 10s 的线性调频信号。

其实现的 MATLAB 程序代码如下：

```
clear all;
t=10e-6;
fs=100e6;
fc=10e6;
B=2e6;
ft=0:1/fs:t-1/fs;
N=length(ft);
k=B/fs*2*pi/max(ft);
y=modulate(ft,fc,fs,'fm',k);
y_fft_result=fft(y);
figure;
subplot(211);plot(ft,y);
xlabel('单位:s');ylabel('单位:V');
title('线性调频信号 y(t)');
subplot(212);plot((0:fs/N:fs/2-fs/N),abs(y_fft_result(1:N/2)));
xlabel('频率  f(单位:Hz)');
title('线性调频信号 y(t)的频谱');
```

运行程序，效果如图 11-2 所示。

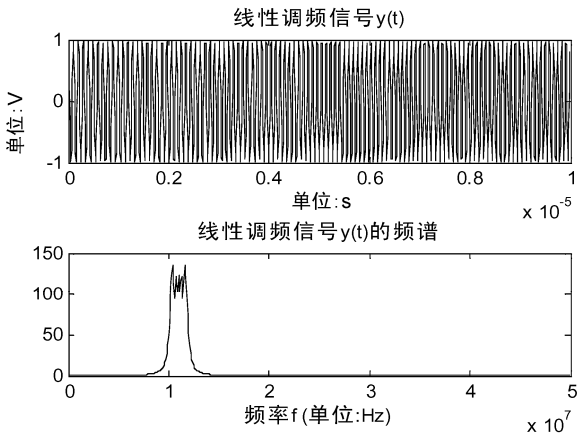


图 11-2　载波 10MHz，带宽 2MHz 的线性调频信号及其频谱

如果要产生初始相位不是 0 的线性调频信号，则不能调用 modulate 函数，因为 modulate 函数产生出来的线性调频信号的初始频率固定为 0 相位。

11.1.3　相位编码信号

线性调频信号、非线性调频信号的调制函数是连续的，属于“连续型”信号；而相位编码信号，其相位调制函数是离散的有限状态，属于“离散型”编码脉冲压缩信号。由于相位编码采用伪随机序列，因此这类信号也称为伪随机编码信号。

数字相位编码调制是利用载波相位的变化来表达数字信号信息的一种调制方法，也叫相移键控。常用的相位编码调制信号有二相码（2PSK）和四相码（4PSK）。

在二相码中，通常用两个相反的相位“0° ”或“180° ”来表示数字信息“0”或者“1”。因此二相码（2PSK）可以表示为

$$S_{2\mathrm{PSK}}(t) = A\cos(\omega_0 + \varphi) \qquad (11\text{-}3)$$

式中：φ由数字码“1”或“0”来决定是“0° ”还是“180° ”，当然φ并不是说只能用“0° ”或“180° ”来表示，也可以用“90° ”还是“270° ”表示。

四相码与二相码基本是一样的，只不过四相码采用 4 个正交的相位来表示 4 个不同的数字信息。在雷达系统中，相位编码信号采用得比较多的是二相码，四相码在通信领域内应用得较多。常用的二相编码信号有 m 序列、L 序列、双素数序列、巴克码序列等。

【例 11-3】产生 7 位巴克码编码的二相码，采样率 100MHz，载波 10MHz，码宽 0.5μs。

其实现的 MATLAB 程序代码如下：

```
clear all;close all;
co=[1 1 1 0 0 1 0];
ta=0.5e-6;
fc=10e6;
fs=100e6;
t_ta=0:1/fs:ta-1/fs;
n=length(co);
pha=0;
t=0:1/fs:7*ta-1/fs;
s=zeros(1,length(t));
for i=1:n
    if co(i)==1
        pha=1;
    else
        pha=0;
    end
    s(1,(i-1)*length(t_ta)+1:i*length(t_ta))=cos(2*pi*fc*t_ta+pha);
end
figure;plot(t,s);
xlabel('t(单位:s)');title('二相码(7 位巴克码)');
```

运行程序，效果如图 11-3 所示。

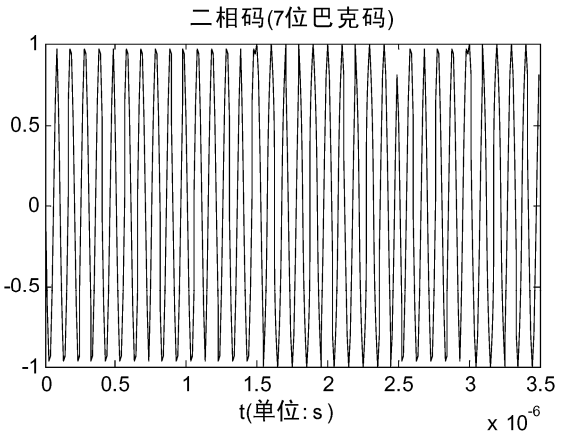


图 11-3　二相码（7 位巴克码）波形图

11.1.4　相位编码脉内线性调频混合调制信号

二相码编码信号对多普勒频率比较敏感，只适用于多普勒频率比较小的场合，但是由于其优越的抗截获性能，常常与线性调频信号组合起来，用于各种低频获雷达系统中。需要说明的是，纯粹的 13 位巴克码信号是没有实用意义的，只适合理论分析，在实际中可以用组合巴克码方式作为脉冲压缩信号。

混合信号是结合调频信号和相位编码信号的优点，把两种信号组合起来，在相位编码信号的每个码元内再进行线性调频信号调制而形成的一种新型适用于脉冲压缩雷达的信号。混合信号对多普勒信号基本不敏感，只是增益略有下降，也没有产生明显的峰值偏移现象，具有相位编码和线性调频两种信号的优点，又能弥补两种信号各自的不足。

设线性调频信号的数字表达式为

$$u_L(t)=\frac{1}{\sqrt{t}}\exp(\mathrm{j}\pi kt^2)[\varepsilon(t)-\varepsilon(t-T)] \tag{11-4}$$

二相编码的脉冲函数为

$$u_B(t)=\frac{1}{\sqrt{P}}\sum_{m=0}^{P-1}c_m\delta(t-mT) \tag{11-5}$$

式中：$\varepsilon(t)$为阶跃函数；$\delta(t)$为冲激函数；T 为子脉冲宽度；P 为码长；k 为线性调频调制斜率；c_m 为一随机序列，取$\{c_m=\pm1\}$；混合脉冲倍为线性调频与二相编码脉冲函数的卷积形式。

利用 MATLAB 产生混合调制信号的方法和产生二相码的方法基本相似，差别在于二相码是单片信号，而混合调制信号脉内是线性调频信号。

【例 11-4】产生 7 位巴克码和线性调频的混合调制信号，码元宽度为 10μs，线性调频的起始频率为 500kHz，调频带宽为 1MHz。

其实现的 MATLAB 程序代码如下：

```
clear all;close all;
co=[1 1 1 0 1 0 1];
ta=10e-6;
fc=0.5e6;
fs=10e6;
t_ta=0:1/fs:ta-1/fs;
```

```
N=length(t_ta);
B=1e6;
k=B/fs*2*pi/max(t_ta);
n=length(co);
pha=0;
s=zeros(1,n*N);
for i=1:n
    if co(i)==1
        pha=1;
    else
        pha=0;
    end
    s(1,(i-1)*N+1:i*N)=cos(2*pi*fc*t_ta+k*cumsum(t_ta)+pha);
end
t=0:1/fs:7*ta-1/fs;
figure;subplot(211);
plot(t,s);xlabel('t(单位:s)');
title('混合调制信号(7 位巴克码+线性调频)');
y_fft_result=abs(fft(s(1:N)));
subplot(212);plot((0:fs/N:fs/2-fs/N),abs(y_fft_result(1:N/2)));
xlabel('频率 f(单位:Hz)');title('码内信号频谱');
```

运行程序，效果如图 11-4 所示。

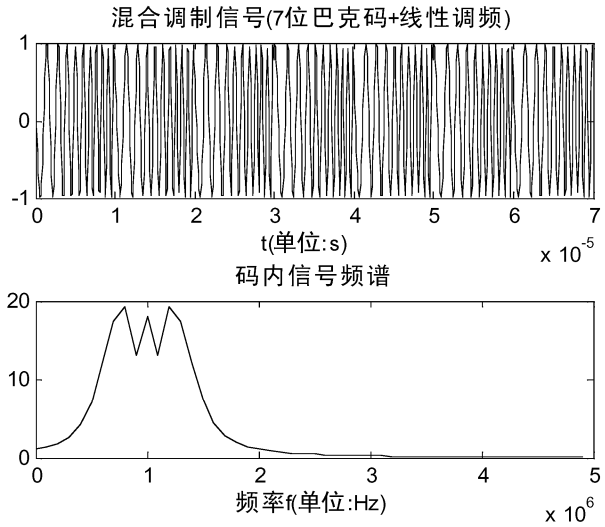


图 11-4　7 位巴克码和线性调频的混合调制信号及脉冲内信号的频谱

11.2　噪声和杂波的产生

在实际的雷达回波信号中，不仅仅有目标的反射信号，同时还有接收机的热噪声、地物杂波、气象杂波等各种噪声和杂波的叠加。由于噪声和杂波都不是确知信号，只能通过统计特性来分析。

11.2.1 随机热噪声

随机热噪声有多种，常见的有概率密度函数服从均匀分布、高斯分布、指数分布等的热噪声。

1．服从均匀分布的热噪声

(*a*–*b*)均匀分布的概率密度函数为

$$p(x)=\frac{1}{b-a} \tag{11-6}$$

根据(*a*–*b*)均匀分布的概率密度函数和(0–1)均匀分布的概率密度函数，可以推导出它们之间的关系为

$$u=-\frac{\xi-a}{a-b} \text{ 或 } \xi=(b-a)\times u+a \tag{11-7}$$

式中：u 为服从(0–1)单位均匀分布；ξ 为服从(*a*–*b*)均匀分布。

所以，根据式（11-7），可以先产生一个服从(0–1)的单位均匀分布的信号，然后再将经过上式的变换，就可以得到一个服从(*a*–*b*)均匀分布的信号了。

MATLAB 自带了（0–1）单位均匀分布的内部函数 rand。其调用格式如下：

```
r = rand(n)
rand(m,n)
rand([m,n])
rand(m,n,p,...)
rand([m,n,p,...])
rand
rand(size(A))
r = rand(..., 'double')
r = rand(..., 'single')
```

【例 11-5】利用 rand 函数产生服从(*a*–*b*)均匀分布的随机序列。

其实现的 MATLAB 程序代码如下：

```
clear all; close all;
a=2;   %(a-b)均匀分布下限
b=3;   %(a-b)均匀分布上限
fs=1e7;   %采样率,单位:Hz
t=1e-3;   %随机序列长度,单位:s
n=t*fs;
rand('state',0);   %把均匀分布伪随机发生器置为 0 状态
u=rand(1,n);   %产生(0-1)单位均匀信号
x=(b-a)*u+a;   %广义均匀分布与单位均匀分布之间的关系
subplot(211);plot(x);   %输出信号图
title('均匀分布信号');
subplot(212);hist(x,a:0.02:b); %输出信号的直方图
title('均匀分布信号直方图');
```

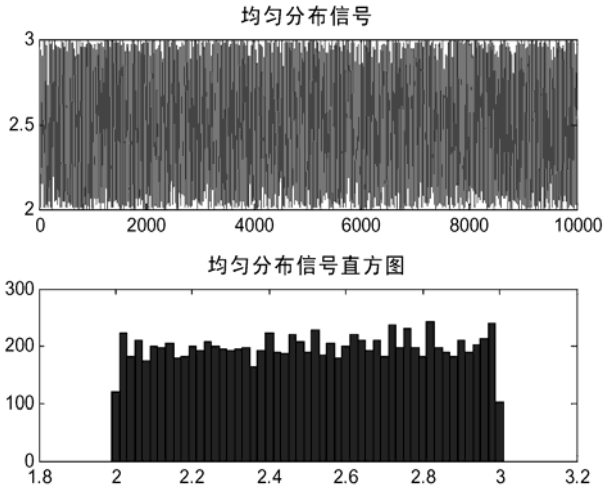


图 11-5　服从(*a*–*b*)均匀分布的随机序列及其直方图

运行程序效果如图 11-5 所示。

2．服从高斯分布的热噪声

均值为 x_0 的高斯分布的概率密度函数为

$$p(x)=\frac{1}{\sqrt{\pi}\sigma}\exp\left(-\frac{(x-x_0)^2}{2\sigma^2}\right) \tag{11-8}$$

MATLAB 本身也自带了标准高斯分布的内部函数 randn。其调用格式如下：

r = randn(n)
randn(m,n)
randn([m,n])
randn(m,n,p,...)
randn([m,n,p,...])
randn(size(A))
r = randn(..., 'double')
r = randn(..., 'single')

【例 11-6】利用 randn 产生一个服从高斯分布的随机序列。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
y=randn(1000);
figure;plot(y);
title('服从高斯分布的随机序列信号');
```

运行程序，效果如图 11-6 所示。

3．服从瑞利分布的热噪声

瑞利（Rayleigh）分布的概率密度函数为

$$p(x)=\begin{cases}\dfrac{x}{\sigma^2}\mathrm{e}^{-\frac{x^2}{2\sigma^2}}, & x\geqslant 0\\ 0, & x<0\end{cases} \tag{11-9}$$

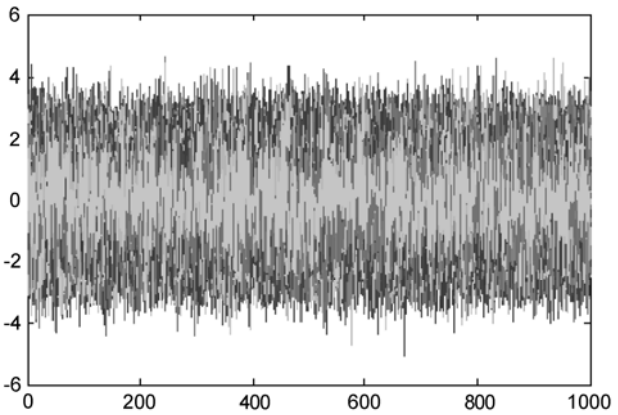


图 11-6　服从高斯分布的随机序列效果图

根据瑞利分布的概率密度函数和(0－1)单位均匀分布的概率密度函数，可以推导出它们之间的关系为

$$\xi_i = \sigma \bullet \sqrt{2 \times \ln \frac{1}{u_i}} \tag{11-10}$$

式中：u 服从(0–1)单位均匀分布；ξ 服从瑞利分布。

所以根据式（11-10），可以先产生一个服从(0－1)分布的信号，然后再将其经过上式的变换，就可以得到一个服从瑞利分布的信号了。

【例 11-7】产生瑞利分布的热噪声。

其实现的 MATLAB 程序代码如下：

```
clear all; close all;
sigma=2;                        %瑞利分布参数 sigma
fs=1e7;                         %采样率,单位:Hz
t=1e-3;                         %随机序列长度,单位:s
t1=0:1/fs:t-1/fs;
n=length(t1);
rand('state',0);                %把均匀分布伪随机发生器置为 0 状态
u=rand(1,n);                    %产生(0-1)单位均匀信号
x=sqrt(2*log2(1./u))*sigma;     %广义均匀分布与单位均匀分布之间的关系
subplot(211);plot(x);           %输出信号图
xlabel('t(单位:s)');
title('瑞利分布信号');
subplot(212);hist(x,0:0.2:20);  %输出信号的直方图
title('瑞利分布信号直方图');
```

运行程序，效果如图 11-7 所示。

4．服从指数分布的热噪声

参数为λ的指数分布的概率密度函数为

$$p(x) = \lambda \mathrm{e}^{-\lambda x} \tag{11-11}$$

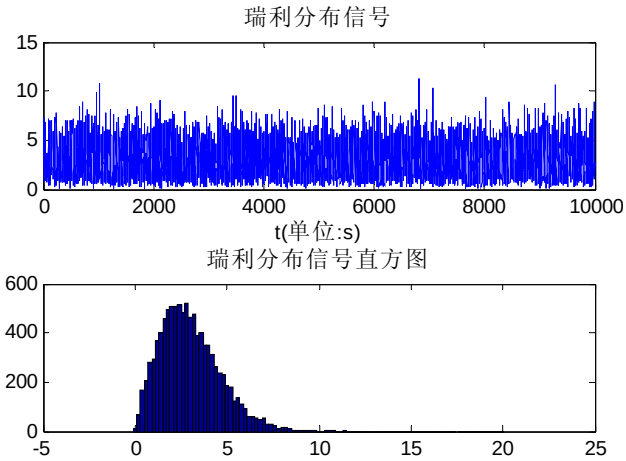


图 11-7　服从瑞利分布热噪声及其直方图

根据指数分布的概率密度函数和 (0 −1) 单位均匀分布的概率密度函数，可以推导出它们之间的关系为

$$u = 1 - \mathrm{e}^{-\lambda\xi} \text{ 或 } \xi_i = -\frac{1}{\lambda}\ln(1-u_i) \qquad (11\text{-}12)$$

由于 u_i 服从(0−1)单位均匀分布，所以(1−u_i)仍然服从(0−1)单位均匀分布，所以式（11-12）可以简化为

$$\xi_i = -\frac{1}{\lambda}\ln u_i \qquad (11\text{-}13)$$

式中：u 服从(0−1)单位均匀分布；ξ服从参数为λ的指数分布。

根据上式，可以先产生一个服从(0−1)单位分布的信号，然后再将其经过上式的变换，就可以得到一个服从参数为λ的指数分布的信号了。

【例 11-8】服从指数分布的热噪声随机序列的实现。

其实现的 MATLAB 程序代码如下：

```
clear all; close all;
dba=2.5;                          %指数分布参数
fs=1e7;                           %采样率,单位:Hz
t=1e-3;                           %随机序列长度,单位:s
n=t*fs;
rand('state',0);                  %把均匀分布伪随机发生器置为 0 状态
u=rand(1,n);                      %产生(0-1)单位均匀信号
x=log2(1-u)/(-dba); %广义均匀分布与单位均匀分布之间的关系
subplot(211);plot(0:1/fs:t-1/fs,x);   %输出信号图
xlabel('t(单位:s)');
title('指数分布信号');
subplot(212);hist(x,0:0.05:4);        %输出信号的直方图
title('指数分布信号直方图');
```

运行程序，效果如图 11-8 所示。

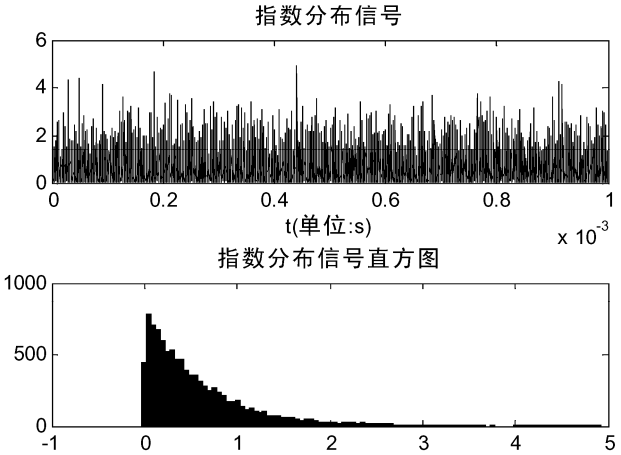


图 11-8　指数分布随机序列及其直方图

11.2.2　杂波的模拟与实现

雷达工作环境的杂波，如地面、海面及空中的云雨、干扰箔条等对雷达性能影响很大。很显然，雷达工作环境的不同，雷达所接收的杂波就不同。

按照杂波背景的不同，通常人们把杂波分为地物杂波、海杂波和气象杂波等类型。不同的杂技类型具有不同的杂波特性。对于地物杂波，可采用幅度概率分布为端利分布、对数正态分布、Weibull 分布的模型来描述，功率谱为高斯谱、立方谱，常用的普型为高斯谱；海杂波可采用幅度为对数正态分布、K 分布的高斯谱杂波模型来表示；气象杂波可采用幅度分布为瑞利分布的高斯谱模型来描述。具体对应某种杂波，采用何种幅度分布及功率谱模型应根据实际的情况而定。

统计模型的杂波模拟就是产生同时具有特定的概率密度和功率谱密度（或者相关函数）的随机序列。产生特定概率分布和任意功率函数的杂波序列的方法有很多，其中较为经典的两种方法是球形不变随机过程法（Spherically Invariant Random Process，SIRP）和广义维纳过程的零记忆非线性变换法（Zero Memory Nonlinearity，ZMNL）。

1．瑞利分布杂波

瑞利分布杂波是雷达杂波中最常用的也是用得最早的一种统计模型。在雷达可分辨范围内，当散射体的数目非常多时，根据散射反射信号振幅和相位的随机特性，它们合成回波的包络振幅是服从瑞利分布的。如果采用 x 表示瑞利分布杂波回波的包络振幅，其下的概率密度函数可表示为

$$p(x)=\begin{cases}\dfrac{x}{\sigma^2}\mathrm{e}^{-\frac{x^2}{2\sigma^2}}, & x\geqslant 0\\ 0, & x<0\end{cases} \qquad (11\text{-}14)$$

式中：σ是杂波的标准差。

【例 11-9】杂波方差为 σ_v=1.0m/s，波长为 5cm，由此 σ_f =40Hz，雷达重复频率为 1000Hz，概率密度参数为 1.2。滤波器的设计采用傅里叶级数展开法，功率谱采用高斯谱模型。模拟的杂波的功率谱密度采用 Burg 法估计得到，概率密度函数的估计采用直方图估计法。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
azi_num=2000; fr=1000;
lamda0=0.005; sigmav=1.0;
sigmaf=2*sigmav/lamda0;
rand('state',sum(100*clock));
d1=rand(1,azi_num);
rand('state',7*sum(100*clock)+3);
d2=rand(1,azi_num);
xi=2*sqrt(-2*log(d1)).*cos(2*pi*d2);
xq=2*sqrt(-2*log(d1)).*sin(2*pi*d2);
coe_num=12;
for n=0:coe_num
    coeff(n+1)=2*sigmaf*sqrt(pi)*exp(-4*sigmaf^2*pi^2*n^2/fr^2)/fr;
end
for n=1:2*coe_num+1
    if n<=coe_num+1
        b(n)=1/2*coeff(coe_num+2-n);
    else
        b(n)=1/2*coeff(n-coe_num);
    end
end
% 生成高斯谱杂波
xxi=conv(b,xi);
xxq=conv(b,xq);
xxi=xxi(coe_num*2+1:azi_num+coe_num*2);
xxq=xxq(coe_num*2+1:azi_num+coe_num*2);
xisigmac=std(xxi);
ximuc=mean(xxi);
yyi=(xxi-ximuc)/xisigmac;
xqsigmac=std(xxq);
xqmuc=mean(xxq);
yyq=(xxq-xqmuc)/xqsigmac;
sigmac=1.2;          %杂波的标准差
yyi=sigmac*yyi;      %使瑞利分布杂波
yyq=sigmac*yyq;
ydata=yyi+j*yyq;
figure;
subplot(2,1,1);plot(real(ydata));
title('瑞利杂波时域波形,实部');
subplot(2,1,2);plot(imag(ydata));
title('瑞利杂波时域波形,虚部');
num=100;             %求概率密度函数的参数
maxdat=max(abs(ydata));
mindat=min(abs(ydata));
NN=hist(abs(ydata),num);
xpdf1=num*NN/((sum(NN))*(maxdat-mindat));
```

```
xaxis1=mindat:(maxdat-mindat)/num:maxdat-(maxdat-mindat)/num;
th_val=(xaxis1./sigmac.^2).*exp(-xaxis1.^2./(2*sigmac.^2));
figure;plot(xaxis1,xpdf1);
hold on; plot(xaxis1,th_val,'r:');
title('杂波幅度分布');
xlabel('幅度'); ylabel('概率密度');
signal=ydata;
signal=signal-mean(signal);
figure; M=256;
psd_dat=pburg(real(signal),32,M,fr);
psd_dat=psd_dat/(max(psd_dat));
freqx=0:0.5*M;
freqx=freqx*fr/M;
plot(freqx,psd_dat); title('杂波频谱');
xlabel('频率/Hz'); ylabel('功率谱密度');
powerf=exp(-freqx.^2/(2*sigmaf.^2));
hold on; plot(freqx,powerf,'r:');
```

运行程序，效果如图 11-9～图 11-11 所示。

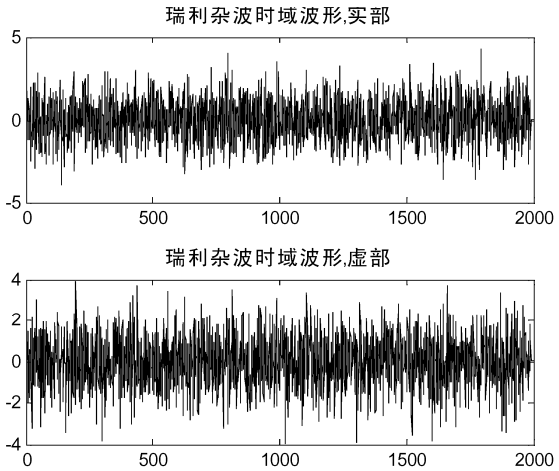


图 11-9　瑞利分布杂波虚实部效果

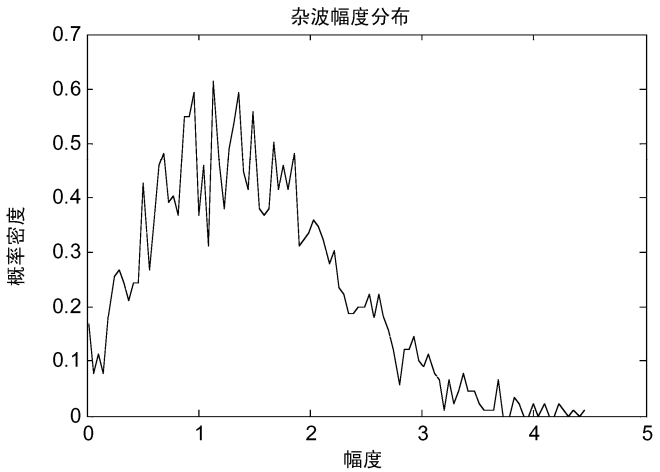


图 11-10　瑞利杂波幅度效果

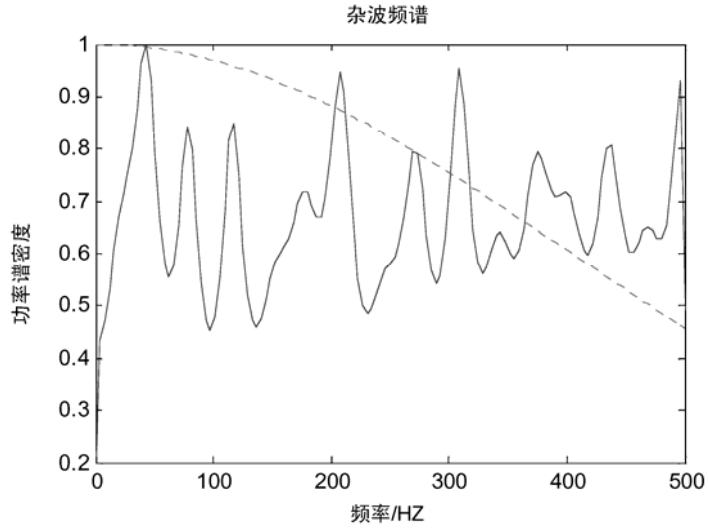


图 11-11　瑞利分布频谱

2．相关对数正态分布杂波

在高分辨率和低擦地角条件下，海面和地面的回波可以认为服从对数正态分布。对数正态分布的概率密度函数为

$$p(x)=\frac{1}{\sqrt{2\pi\sigma_c x}}\exp\left[-\ln^2\left(\frac{x}{\mu_c}/2\sigma_c^2\right)\right] \qquad (11\text{-}15)$$

式中：μ_c 是尺度参数，表示分布的中位数；σ_c 是形状参数，表明分布的偏斜度。

【例 11-10】概率密度参数σ_c=0.6dB，尺度参数μ_c=10，谱型为高斯谱，其σ_f为 40Hz，滤波器的设计采用傅里叶级数展开法，模拟的杂波的功率谱密度采用 Burg 法估计得到，概率密度函数的估计采用直方图估计法。

其实现的 MATLAB 程序代码如下：

```
>>    clear all;
azi_num=2000; fr=1000;
lamda0=0.05; sigmav=1.0;
sigmaf=2*sigmav/lamda0;
rand('state',sum(100*clock));    %产生服从 U(0-1)的随机序列
d1=rand(1,azi_num);
rand('state',7*sum(100*clock)+3);
d2=rand(1,azi_num);
xi=2*sqrt(-2*log(d1)).*cos(2*pi*d2);        %正交且独立的高斯序列~N(0,1)
coe_num=12;                                 %求滤波器系数,用傅里叶级数展开法
for n=0:coe_num
      coeff(n+1)=2*sigmaf*sqrt(pi)*exp(-4*sigmaf^2*pi^2*n^2/fr^2)/fr;
end
for n=1:2*coe_num+1
      if n<=coe_num+1
            b(n)=1/2*coeff(coe_num+2-n);
      else
```

```
            b(n)=1/2*coeff(n-coe_num);
        end
    end
    % 生成高斯谱杂波
    xxi=conv(b,xi);
    xxi=xxi(coe_num*2+1:azi_num+coe_num*2);
    xisigmac=std(xxi);
    ximuc=mean(xxi);
    yyi=(xxi-ximuc)/xisigmac;
    muc=10;                          %中位值
    sigmac=0.6;                      %形状参数
    yyi=sigmac*yyi+log(muc);
    xdata=exp(yyi);                  %参数正态分布的杂波序列
    figure;
    subplot(2,1,1);plot(xdata);
    title('对数正态杂波时域波形');
    num=100;
    maxdat=max(abs(xdata));
    mindat=min(abs(xdata));
    NN=hist(abs(xdata),num);
    xpdf1=num*NN/((sum(NN))*(maxdat-mindat));   %用直方图估计的概率密度函数
    xaxis1=mindat:(maxdat-mindat)/num:maxdat-(maxdat-mindat)/num;
    th_val=lognpdf(xaxis1,log(muc),sigmac);
    subplot(2,2,3);plot(xaxis1,xpdf1);
    hold on; plot(xaxis1,th_val,'r:');
    title('杂波幅度分布');
    xlabel('幅度'); ylabel('概率密度');
    signal=xdata;
    signal=signal-mean(signal);          %求功率谱密度,先去掉直流分量
    M=128;
    psd_dat=pburg(real(signal),16,M,fr);
    psd_dat=psd_dat/(max(psd_dat));     %归一化
    freqx=0:0.5*M;
    freqx=freqx*fr/M;
    subplot(2,2,4);plot(freqx,psd_dat);
    title('杂波频谱');
    xlabel('频率/Hz'); ylabel('功率谱密度');
    powerf=exp(-freqx.^2/(2*sigmaf.^2));
    hold on; plot(freqx,powerf,'r:');
```

运行程序，效果如图 11-12 所示。

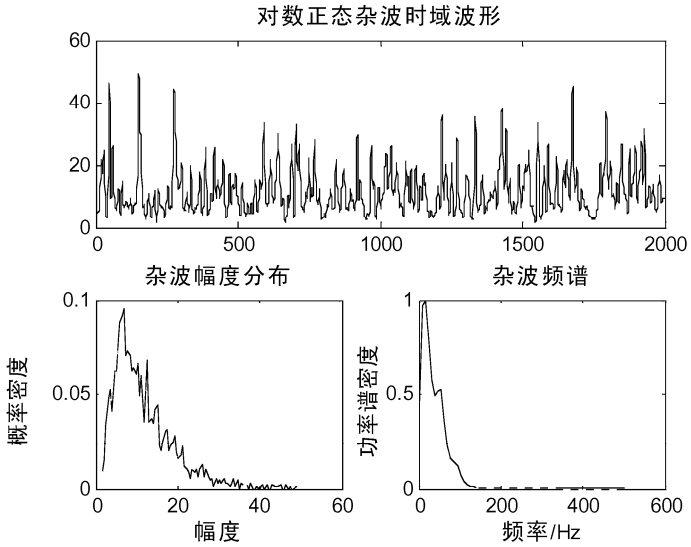


图 11-12　相关对数正态分布杂波及其概率分布密度和频谱仿真结果

11.3　小波在语音信号处理中的应用

语音识别与合成技术是一种人机语言通信技术，属于计算机智能接口技术。计算机智能接口技术主要包括计算机听觉和视觉。多媒体技术也主要是利用计算机语音处理和图像处理的能力为人们提供一种更为方便的人机界面。使人与计算机之间、人与人之间的通信更加方便。

语音识别技术的应用，本质上是基于它能将人的语音转化为语言代码。语音是语言信息的载体，语音识别的基本任务是将输入的语音转化为相应的语言代码（如文字或词语的代号）。这样，不仅在存储或传输这样的语言代码时的数码率比起存储原来的语音信号来大幅度降低，还在于它把一种连续的语音信号变成了一种有限符号集中的符号（或代码），这样的符号容易被计算机（或专用信息处理单元）理解其含义，而且便于与人进行交流，因而可以进行十分广泛的应用。

语音合成的应用已经在许多方面推向了实际应用，如公共交通中的自动报站、各种场合的自动报时、自动告警、电话自动查询服务、文本校对中的语音提示等。

语音识别主要是在近 20 年中发展起来的，现在已在安全加密、银行信息电话查询服务等方面得到了应用，此外在公安机关破案和法庭取证方面可以得到应用。

语种辨识在军事情报工作、国家安全事务中有重要应用。

语音编码技术的应用价值是不须多说的，它的根本作用是使语音通信数字化，而语音通信的数字化将使通信技术的水平提高一大步。目前正在蓬勃兴起的移动通信和个人通信，语音编码技术就是其中非常的支撑技术。

目前小波在语音信号处理中的应用非常广泛，例如在语音识别的信号预处理、语音端点检测、语音分析与合成、语音增强以及语音编码等中都有非常广泛的应用。

11.3.1　小波在语音信号增强中的应用

语音增强目的是从带噪语音中提取尽可能纯净的原始语音，即在前端消除含噪语音信号中

的噪声成分，提高输入信号的信噪比。在实际应用环境中，语音都会不同程度受到环境噪声的干扰，噪声会影响语音质量，严重的情况下将语音完全淹没在噪声中，无法分辨。同时，语音质量的下降也会使许多语音处理系统的性能急剧恶化。语音增强技术无论在日常生活中，还是在军事领域，或者对语音处理技术本身来说都很有应用价值。

小波的多分辨分析特性使其在非平稳信号以及图像等的去噪中都得到了广泛的应用，因此小波同样可以应用于语音信号的去噪中。

【例 11-11】在读入的语音信号中加入正态随机噪声，然后对含噪声的语音信号进行小波分解，估计噪声的方差，然后获取去噪的阈值并对小波分解的高频系数进行阈值量化，得到去噪后的语音信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
% 在噪声环境下语音信号的增强
sound=wavread('Blip.wav');   %语音信号的读入
cound=length(sound);
noise=0.05*randn(1,cound);
y=sound'+noise;
% 用小波函数'db6'对信号进行 3 层分解
[C,L]=wavedec(y,3,'db6');
% 估计尺度 1 的噪声标准偏差
sigma=wnoisest(C,L,1);
alpha=2;
% 获取消噪过程中的阈值
thr=wbmpen(C,L,sigma,alpha);
keepapp=1;
% 对信号进行消噪
yd=wdencmp('gbl',C,L,'db6',3,thr,'s',keepapp);
subplot(1,2,1); plot(sound);
title('原始语音信号');
subplot(1,2,2);plot(yd);
title('去噪后的语音信号');
```

运行程序，效果如图 11-13 所示。

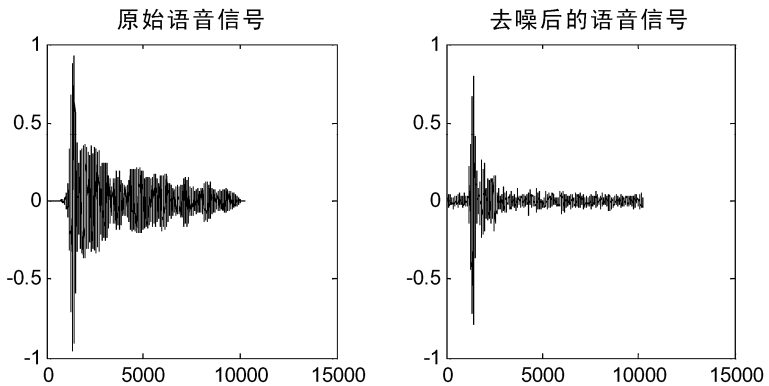


图 11-13　小波应用于语音增强

由结果可以看出，这种方法能有效地去除噪声，还原了原始语音信号。

【例 11-12】将读入的语音信号加入正态随机噪声，然后对含噪声的语音信号进行小波分解，再获取去噪的阈值并对小波分解的高频系数进行阈值量化，得到去噪后的语音信号。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
% 在噪声环境下语音信号的增强
sound=wavread('Blip.wav');  %语音信号的读入
cound=length(sound);
noise=0.05*randn(1,cound);
y=sound'+noise;
% 获取噪声的阈值
[thr,sorh,keepapp]=ddencmp('den','wv',y);
% 对信号进行消噪
yd=wdencmp('gbl',y,'db4',2,thr,sorh,keepapp);
subplot(1,2,1); plot(sound);
title('原始语音信号');
subplot(1,2,2);plot(yd);
title('去噪后的语音信号');
```

运行程序，效果如图 11-14 所示。

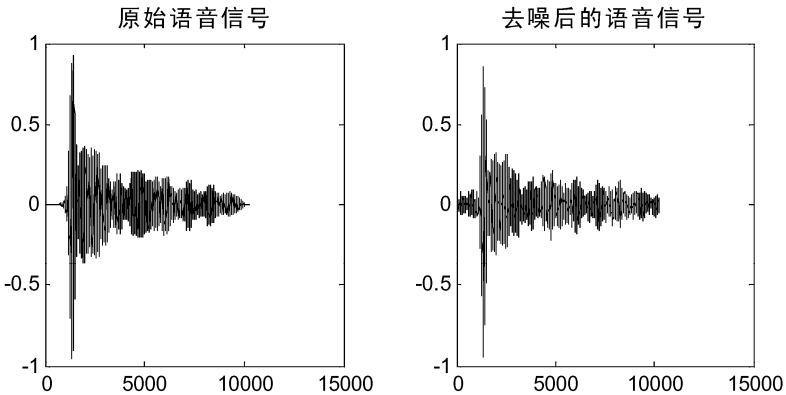


图 11-14　语音增强

11.3.2　小波在语音信号压缩中的应用

小波在语音信号压缩中的主要应用是在编码前对信号进行小波变换并对系数进行处理。一般步骤是将信号进行小波分解，然后选择阈值对小波系数进行压缩，最后使用编码算法对处理后的小波系数进行编码实现信号的压缩编码。

【例 11-13】对语音信号进行小波分解后获取语音信号压缩的阈值，最后利用 wdencmp 进行语音信号的压缩。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%语音信号的读入
sound=wavread('Blip.wav');
% 用小波函数 haar 对信号进行 3 层分解
[C,L]=wavedec(sound,3,'haar');
```

```
alpha=1.5;
% 获取信号压缩的阈值
[thr,nkeep]=wdcbm(C,L,alpha);
% 对信号进行压缩
[cp,cxd,lxd,per1,per2]=wdencmp('lvd',C,L,'haar',3,thr,'s');
subplot(1,2,1); plot(sound);
title('原始语音信号');
subplot(1,2,2);plot(cp);
title('压缩后的语音信号');
```

运行程序，效果如图 11-15 所示。

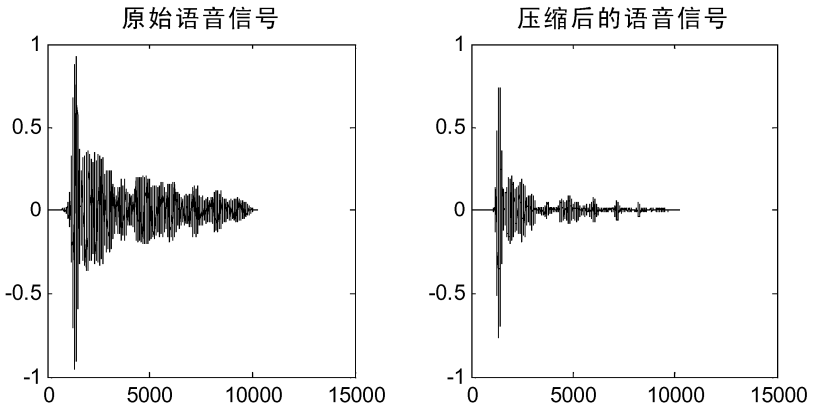


图 11-15　语音压缩信号效果

【例 11-14】与前面的处理方法相同，除了使用 wdcbm 获取信号压缩的阈值外，也可以使用本例中的 ddencmp 函数获取压缩阈值。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%语音信号的读入
sound=wavread('Blip.wav');
% 用小波函数 haar 对信号进行 5 层分解
[C,L]=wavedec(sound,5,'haar');
% 获取信号压缩的阈值
[thr,nkeep]=ddencmp('cmp','wv',sound);
% 对信号进行压缩
cp=wdencmp('gbl',C,L,'haar',5,thr,'s',1);
subplot(1,2,1); plot(sound);
title('原始语音信号');
subplot(1,2,2);plot(cp);
title('压缩后的语音信号');
```

运行程序，效果如图 11-16 所示。

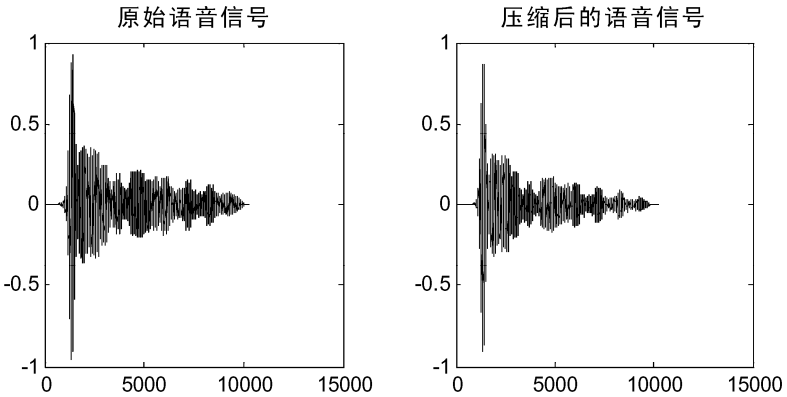


图 11-16　语音压缩效果

参 考 文 献

[1] 程相君，陈生潭.信号与系统. 西安：西安电子科技大学出版社，1990.

[2] 奥本海姆 A V，谢费 R W. 数字信号处理. 北京：科学出版社，1992.

[3] 张志涌. 掌握与精通 MATLAB. 北京：北京航空航天大学出版社，1998.

[4] 刘卫国. 科学计算与 MATLAB 语言. 北京：中国铁道出版社，2000.

[5] 王立宁，乐光新，詹菲. MATLAB 与通信仿真. 北京：人民邮电出版社，2000.

[6] 徐昕，李涛，等. MATLAB 工具箱应用指南——控制工程篇. 北京：电子工业出版社，2000.

[7] 放许波. MATLAB 工程数学应用. 北京：清华大学出版社，2000.

[8] 向敬成，张明友. 雷达系统. 北京：电子工业出版社，2001.

[9] 何强，何英. MATLAB 扩展编程. 北京：清华大学出版社，2002.

[10] 张铮，杨文平，石博强，等. MATLAB 程序设计与实例应用. 北京：中国铁道出版社，2003

[11] 万建伟. 信号处理仿真技术. 长沙：国防科技大学出版社，2003.

[12] 薛定宇，陈阳泉. 高等应用数学问题的 MATLAB 求解. 北京：清华大学出版社，2004.

[13] 苏金明，王永利. MATLAB 7.0 实用指南上册. 北京：电子工业出版社，2004.

[14] 薛年喜. MATLAB 在数字信号处理中的应用. 北京：清华大学出版社，2004.

[15] 陈怀琛，吴大正，高西全，等. MATLAB 语言及其在电子信息工具中的应用. 北京：清华大学出版社，2004.

[16] 张瑞丰. 精通 MATLAB 6.5. 北京：中国水利水电出版社，2004.

[17] 张旭东，卢国栋，冯键. 图像编码基础和小波压缩技术——原理、算法和标准. 北京：清华大学出版社，2004.

[18] 罗军辉. MATLAB 7.0 在数学信号处理中的应用. 北京：机械工业出版社，2005.

[19] 张照明，刘政波，刘斌. 应用 MATLAB 实现信号分析和处理. 北京：科学出版社，2005.

[20] 沈邦兴，文昌俊. 实验设计与工程应用. 北京：中国计量出版社，2005.

[21] 刘卫国. MATLAB 程序设计与应用（第二版）. 北京：高等教育出版社，2006.

[22] 郭仕剑，王宝顺，贺志国，等. MATLAB 7.X 数学信号处理. 北京：人民邮电出版社，2006.

[23] 万永革. 数字信号处理的 MATLAB 实现. 北京：科学出版社，2006.

[24] 唐向宏，岳恒立，郑雪峰. MATLAB 及在电子信息类课程中的应用. 北京：电子工业出版社，2006.

[25] 甘俊英，胡异丁. 基于 MATLAB 的信号与系统实验指导. 北京：清华大学出版社，2006.