国家自然科学基金资助项目 50278062

# LINGO 8.0 及其在环境系统优化中的应用

张宏伟　牛志广　编著

天津大学出版社
TIANJIN UNIVERSITY PRESS

## 内容提要

LINGO 8.0 是由美国 LINDO Systems 公司开发的系列最优化软件之一。该软件具有强大的模型表达和求解能力，操作简便，界面友好，深受广大科研工作者的喜爱。本书首先介绍了 LINGO 模型语言的基本知识，包括集合、变量、域、运算符和函数等，然后对 LINGO 软件的窗口命令和命令行命令作了详细的说明，接着又介绍了 LINGO 软件的一些外部接口技术，包括与外部文件、电子数据表、数据库以及其他应用程序等的接口，最后利用 LINGO 建立和求解了一些环境系统优化方面的模型，为这方面的研究开拓了一条新的思路。

本书可作为高等院校环境工程专业、系统工程专业以及其他经管、理工类专业的专科生、本科生以及研究生的教材，也可作为广大科研人员的参考书。

**图书在版编目 (CIP) 数据**

LINGO 8.0 及其在环境系统优化中的应用 /张宏伟，牛广志编著. —天津：天津大学出版社，2005.10
ISBN 7-5618-2211-1

Ⅰ.L… Ⅱ.①张… ②牛… Ⅲ.环境系统 - 最优化算法 - 应用软件，LINGO 8.0 Ⅳ.X21 - 39

**中国版本图书馆 CIP 数据核字 (2005) 第 116037 号**

| | |
|---|---|
| 出版发行 | 天津大学出版社 |
| 出 版 人 | 杨欢 |
| 地　　址 | 天津市卫津路 92 号天津大学内 (邮编：300072) |
| 电　　话 | 发行部：022-27403647　邮购部：022-27402742 |
| 网　　址 | www.tjup.com |
| 印　　刷 | 昌黎太阳红彩色印刷有限责任公司 |
| 经　　销 | 全国各地新华书店 |
| 开　　本 | 185mm × 260mm |
| 印　　张 | 19.5 |
| 字　　数 | 511 千 |
| 版　　次 | 2005 年 10 月第 1 版 |
| 印　　次 | 2005 年 10 月第 1 次 |
| 印　　数 | 1 - 2 500 |
| 定　　价 | 32.00 元 |

# 前　言

LINGO 软件是由美国 LINDO Systems 公司出品的科学计算软件，主要用于求解非线性规划数学模型，在社会的各个领域都有广泛的使用。该软件从 20 世纪 80 年代发展到现在，已经逐步走向成熟，使用起来非常方便，和其他软件的配合能力也日益加强。对于使用者来说，LINGO 的模型语言规则简单，只要具备基本的计算机语言能力便可使用该软件。

运用 LINGO 软件建立和求解像环境系统优化模型这样复杂的模型，可以大大减轻科研工作者研究数学方法的负担。以往，当一个成熟的模型问世以后，往往由于变量、关系式以及约束条件类型众多或形式复杂，使得一般的数学手段不能奏效，所以模型的求解仍需要花费大量的精力和时间，而 LINGO 软件能够很好地协助科研工作者完成模型求解的工作。

本书共有 13 章，可划分为三大部分。第一部分（第 1 章～第 6 章）主要介绍 LINGO 模型语言的基本概念以及 LINGO 软件的基本使用方法，包括 LINGO 的概述，LINGO 模型中集合、变量、数据域和初始化域等组成部分的概念、定义方法等，使用 Windows 版本的 LINGO 软件时需要用到的窗口命令，其他操作系统下 LINGO 软件以及书写 LINGO 脚本文件时用到的命令行命令。第二部分（第 7 章～第 12 章）主要介绍建立和求解 LINGO 模型以及开发用户自己的应用程序时所需的必备知识，其中包括 LINGO 模型中的运算符，函数的类型和使用方法，LINGO 与外部文件、电子数据表、数据库的接口技术，还特别介绍了开发用户在用自己的应用程序时需要用到的 LINGO 与其他应用软件（如 Visual Basic、Visual C + +、Delphi 等）的接口技术，最后简单介绍了数学模型的类型以及在 LINGO 中相应求解方法的特点，帮助用户建立最为适用的模型。第三部分（第 13 章）主要收集了一些在环境系统优化领域中较为经典的例子，并使用 LINGO 建立和求解相应的模型，证明了 LINGO 软件的有效性和实用性。

本书内容全面，涉及 LINGO 模型语言和软件的所有知识，且阐述细致入微，在叙述每一部分内容时均考虑到不同层次用户的需要，力争做到通俗易懂，深入浅出，在环境系统优化方面的应用，不仅能够为环境方面的科研工作者提供新思路，还能为解决其他领域的最优化问题提供参考。

本书可作为高等院校环境工程专业、系统工程专业以及其他经管、理工类专业的专科生、本科生以及研究生的教材，也可作为广大科研人员的参考书。

在本书的编写过程中，得到了美国 LINDO Systems 公司的大力支持，得到了使用其电子帮助文档的许可（试用版本和相应的电子帮助文档可在其网站 www.lindo.com 下载），在此对该公司表示衷心感谢。

在这里需要特别感谢王亮、郭祎萍、孙亚梅、岳琳、李蕾、张永举，他们在文稿翻译、例题验证、模型编写以及成书过程中做了大量的工作，还需要感谢天津大学学报编辑部的王新英老师对我们的指导。

如果本书能够对读者有所帮助，将是我们的荣幸。由于水平有限、经验不足，错误和不当之处在所难免，诚请读者批评指正。

编者

2005 年 8 月于北洋园

# 目　　录


第 1 章　LINGO 概述 …… (1)
1.1　Windows 版本的 LINGO 使用简介 …… (1)
1.1.1　安装 LINGO …… (1)
1.1.2　运行 LINGO …… (1)
1.1.3　LINGO 模型实例 …… (3)
1.1.4　模型的求解 …… (4)
1.1.5　求解状态窗口 …… (5)
1.1.6　模型结果报告 …… (10)
1.1.7　保存 …… (10)
1.2　在其他操作系统中使用 LINGO …… (10)
1.2.1　命令行提示 …… (10)
1.2.2　在命令行中输入模型 …… (11)
1.2.3　在命令行中求解模型 …… (11)
1.2.4　在命令行中打印和保存 …… (12)
1.3　研究模型的结果报告 …… (12)
1.3.1　介绍 …… (12)
1.3.2　Reduced Cost (降低的成本) …… (12)
1.3.3　Slack (松弛变量值) or Surplus (剩余变量值) …… (13)
1.3.4　Dual Price (对偶变量值) …… (13)
1.4　LINGO 模型语言介绍 …… (13)
1.4.1　LINGO 模型语言的特点 …… (13)
1.4.2　运输模型 …… (14)
1.5　LINGO 语言的其他特征 …… (20)
1.5.1　约束条件的命名 …… (21)
1.5.2　模型的标题 …… (22)
1.6　问题的最大规模 …… (23)
第 2 章　集合的使用 …… (24)
2.1　使用集合的目的 …… (24)
2.2　集合简介 …… (24)
2.3　模型的集合域 …… (25)
2.3.1　基本集合的定义 …… (25)
2.3.2　衍生集合的定义 …… (26)
2.3.3　总结 …… (28)
2.4　模型的数据域 …… (28)

2.5 集合循环函数 …… (30)
2.5.1 @SUM 函数 …… (31)
2.5.2 @MIN 和@MAX 函数 …… (31)
2.5.3 @FOR 函数 …… (32)
2.5.4 嵌套的集合循环函数 …… (33)
2.5.5 总结 …… (33)
2.6 基于集合的模型举例 …… (33)
2.6.1 基本集合模型举例——员工安排模型 …… (33)
2.6.2 稠密衍生集合模型举例——配比模型 …… (37)
2.6.3 稀疏衍生集合模型举例——直接列表 …… (40)
2.6.4 稀疏衍生集合模型举例——元素过滤器 …… (44)
2.7 总结 …… (48)
第3章 变量限定函数的使用 …… (49)
3.1 简介 …… (49)
3.2 整型变量 …… (49)
3.2.1 普通整型变量 …… (50)
3.2.2 二进制整型变量 …… (52)
3.3 自由变量 …… (61)
3.3.1 自由变量举例——预测 …… (61)
3.3.2 建立模型 …… (62)
3.3.3 结果 …… (63)
3.4 限界变量 …… (65)
第4章 数据域和初始化域 …… (66)
4.1 模型的数据域 …… (66)
4.1.1 数据域简介 …… (66)
4.1.2 参数 …… (67)
4.1.3 估值分析 …… (67)
4.1.4 把属性初始化为同一个固定值 …… (68)
4.1.5 在数据域中省略值 …… (69)
4.2 模型的初始化域 …… (69)
第5章 窗口命令 …… (71)
5.1 窗口命令的使用 …… (71)
5.1.1 菜单 …… (71)
5.1.2 工具栏 …… (71)
5.1.3 快捷键 …… (72)
5.2 主要窗口命令 …… (72)
5.3 窗口命令详细介绍 …… (74)
5.3.1 File 菜单 …… (74)

5.3.2 Edit 菜单 …… (83)
5.3.3 LINGO 菜单 …… (92)
5.3.4 Window 菜单 …… (122)
5.3.5 Help 菜单 …… (124)
第6章 命令行命令 …… (128)
6.1 命令的简要介绍 …… (128)
6.2 命令的详细介绍 …… (130)
6.2.1 信息命令 …… (131)
6.2.2 输入命令 …… (132)
6.2.3 显示命令 …… (136)
6.2.4 文件输出命令 …… (140)
6.2.5 求解命令 …… (145)
6.2.6 编辑命令 …… (148)
6.2.7 交互参数命令 …… (150)
6.2.8 设置命令 …… (151)
6.2.9 其他命令 …… (152)
第7章 LINGO 的运算符和函数 …… (153)
7.1 标准运算符 …… (153)
7.1.1 算术运算符 …… (153)
7.1.2 逻辑运算符 …… (154)
7.1.3 关系运算符 …… (154)
7.1.4 运算符优先级 …… (155)
7.2 数学函数 …… (155)
7.3 金融函数 …… (156)
7.4 概率函数 …… (156)
7.5 集合处理函数 …… (158)
7.6 输入/输出函数 …… (159)
7.7 其他函数 …… (161)
第8章 与外部文件的接口 …… (164)
8.1 利用复制和粘贴命令传输数据 …… (164)
8.1.1 从 Excel 中导入数据 …… (164)
8.1.2 将数据导出到 Word 文件中 …… (165)
8.2 文本文件接口函数 …… (165)
8.2.1 利用@FILE 函数导入数据 …… (166)
8.2.2 在运输模型中使用@FILE 函数 …… (166)
8.2.3 利用@TEXT 函数导出数据 …… (168)
8.2.4 实例——在员工安排模型中使用@TEXT 函数 …… (169)
8.3 LINGO 命令脚本 …… (171)

8.3.1 一个命令脚本实例 …… (171)
8.3.2 AUTOLG.DAT 脚本文件 …… (173)
8.4 在命令行中的指定文件 …… (173)
8.5 重新定位输入和输出 …… (174)
第9章 与电子数据表的接口 …… (176)
9.1 从电子数据表中导入数据 …… (176)
9.1.1 使用@OLE 函数从 Excel 中导入数据 …… (176)
9.1.2 在运输模型中使用@OLE 函数导入数据 …… (177)
9.2 将求解结果导出到电子数据表中 …… (179)
9.2.1 使用@OLE 函数将求解结果导出到 Excel 中 …… (179)
9.2.2 在运输模型中使用@OLE 函数导出数据 …… (180)
9.2.3 输出统计报告 …… (181)
9.3 利用 OLE 技术实现与 Excel 的自动连接 …… (183)
9.4 将 LINGO 模型嵌入 Excel 中 …… (185)
9.5 在 LINGO 模型中嵌入 Excel 表格 …… (188)
第10章 与数据库的接口 …… (192)
10.1 ODBC 数据源 …… (192)
10.1.1 为 Access 数据库建立 ODBC 数据源 …… (193)
10.1.2 为 Oracle 数据库建立 ODBC 数据源 …… (197)
10.1.3 为 SQL Server 数据库建立 ODBC 数据源 …… (198)
10.2 利用@ODBC 函数从数据库中导入数据 …… (201)
10.2.1 利用@ODBC 函数导入数据语法 …… (201)
10.2.2 在 PERT 模型中利用@ODBC 函数导入数据 …… (202)
10.3 利用@ODBC 函数导出数据 …… (204)
10.3.1 利用@ODBC 函数导出数据的语法 …… (204)
10.3.2 在 PERT 模型中利用@ODBC 函数导出数据 …… (204)
10.3.3 输出统计报告 …… (205)
第11章 与其他应用软件的接口 …… (207)
11.1 LINGO 动态链接库 …… (207)
11.1.1 在员工安排模型中应用 LINGO DLL …… (207)
11.1.2 模型 …… (208)
11.1.3 @POINTER 函数 …… (209)
11.1.4 LINGO DLL 的输出函数 …… (210)
11.1.5 使用 LINGO DLL 和 Visual C++ 求解员工安排模型 …… (214)
11.1.6 使用 LINGO DLL 和 Visual Basic 求解员工安排模型 …… (220)
11.1.7 使用 LINGO DLL 和 Delphi 求解员工安排模型 …… (225)
11.1.8 回调函数 …… (230)
11.1.9 总结 …… (238)

11.2 用户定义函数 …… (238)
11.2.1 在 Windows 系统下安装@USER 函数 …… (239)
11.2.2 Visual C++ 例子 …… (239)
第 12 章 数学模型 …… (243)
12.1 LINGO 内部算法的使用 …… (243)
12.2 约束条件的类型 …… (243)
12.2.1 线性约束 …… (244)
12.2.2 非线性约束 …… (244)
12.3 局部最优和全局最优 …… (245)
12.4 函数的凹凸性 …… (245)
12.5 平滑和非平滑函数 …… (246)
12.6 非线性模型的解决方法 …… (247)
12.6.1 为变量定界 …… (247)
12.6.2 为变量指定初值 …… (247)
12.6.3 确定模型的合理数量级范围 …… (247)
12.6.4 简化关系 …… (247)
12.6.5 减少整型限制 …… (248)
第 13 章 LINGO 在环境系统优化中的应用 …… (249)
13.1 污染企业的生产安排问题 …… (249)
13.1.1 问题描述 …… (249)
13.1.2 LINGO 模型 …… (250)
13.1.3 求解结果 …… (250)
13.2 水处理最优方案问题 …… (251)
13.2.1 问题描述 …… (251)
13.2.2 LINGO 模型 …… (251)
13.2.3 求解结果 …… (252)
13.3 成本—效益问题 …… (253)
13.3.1 问题描述 …… (253)
13.3.2 LINGO 模型 …… (254)
13.3.3 求解结果 …… (254)
13.4 工业废水处理问题 …… (255)
13.4.1 问题描述 …… (255)
13.4.2 LINGO 模型 …… (255)
13.4.3 求解结果 …… (255)
13.5 河流污染控制问题 …… (256)
13.5.1 问题描述 …… (256)
13.5.2 LINGO 模型 …… (257)
13.5.3 求解结果 …… (258)

13.6 水资源开发利用问题 …… (260)
13.6.1 问题描述 …… (260)
13.6.2 LINGO 模型 …… (262)
13.6.3 求解结果 …… (262)
13.7 水资源分配问题 …… (263)
13.7.1 问题描述 …… (263)
13.7.2 LINGO 模型 …… (264)
13.7.3 求解结果 …… (265)
13.8 水库群最优调度问题 …… (266)
13.8.1 问题描述 …… (266)
13.8.2 LINGO 模型 …… (268)
13.8.3 求解结果 …… (269)
13.9 多目标土地规划问题 …… (273)
13.9.1 问题描述 …… (273)
13.9.2 LINGO 模型 …… (275)
13.9.3 求解结果 …… (277)
13.10 用地规划问题 …… (280)
13.10.1 问题描述 …… (280)
13.10.2 LINGO 模型 …… (281)
13.10.3 求解结果 …… (281)
13.11 大气污染物排放的控制问题 …… (282)
13.11.1 问题描述 …… (282)
13.11.2 LINGO 模型 …… (283)
13.11.3 求解结果 …… (284)
13.12 固体废弃物处置问题 …… (286)
13.12.1 问题描述 …… (286)
13.12.2 LINGO 模型 …… (287)
13.12.3 求解结果 …… (287)
13.13 肥料贮存问题 …… (288)
13.13.1 问题描述 …… (288)
13.13.2 LINGO 模型 …… (290)
13.13.3 求解结果 …… (291)
13.14 农业面源污染控制问题 …… (294)
13.14.1 问题描述 …… (294)
13.14.2 LINGO 模型 …… (294)
13.14.3 求解结果 …… (295)
13.14.4 应用 Delphi 建立模型界面 …… (295)
13.15 农药管理问题 …… (298)

13.15.1 问题描述 …… (298)
13.15.2 LINGO 模型 …… (299)
13.15.3 求解结果 …… (300)
参考文献 …… (301)

# 第 1 章　LINGO 概述

由美国 LINDO 公司开发的 LINGO 软件是一个利用线性和非线性最优化方法将复杂的大型规划问题转化为简明公式的工具，具有简单、实用的特点。它可以建立、求解最优化模型并分析所得结果。LINGO 的最优化功能可以帮助用户找到可行解中的最佳结果，从而获得最大的利润、最高的产量、最大的满足等。换句话说，就是以最低的投入、最少的浪费或者最小的不满达到预期的最佳效果。通常利用最优化技术可以使时间、金钱、设备、人力、库存等资源得到最有效的利用。最优化问题分为线性问题和非线性问题，这取决于变量和变量之间的关系是线性的还是非线性的。

在本章中，将首先对 LINGO 软件的功能和用法进行全面而简要的介绍，使用户对 LINGO 软件有大概的认识，能够胜任简单的编程工作。在以后的章节中，对 LINGO 软件的各个部分作详细介绍，从而为高级用户提供使用指导，为建立复杂的 LINGO 模型并以最佳方式进行求解创造条件。

## 1.1　Windows 版本的 LINGO 使用简介

### 1.1.1　安装 LINGO

这一部分内容将介绍如何在 Windows 环境下安装 LINGO 软件。若要在 Windows 以外的平台上安装 LINGO，请参照软件中的安装说明。

安装 LINGO 软件非常简单，运行 LINGO 文件夹中的安装程序 SETUP.exe，LINGO 的安装程序就启动了，并且会指导用户一步一步地将 LINGO 安装在硬盘中。

LINGO 的一些版本要求用户在安装程序完成后输入密码，并且密码只需要在首次启动 LINGO 时输入一次，以后再运行将不再输入。若不知道密码，一般可以在光盘盒上找到。

如果所使用的 LINGO 版本要求输入密码，会在启动 LINGO 时出现图 1-1 所示的对话框。

在编辑框中输入密码（包括连字符，并且注意密码是区分大小写的），然后点击 OK 按钮进行确认，LINGO 就启动了。

如果没有密码，还可以点击 Demo 按钮，在试用模式下运行 LINGO。在试用模式下，LINGO 几乎具有标准版本的所有功能，只是对问题的规模进行了限制。

### 1.1.2　运行 LINGO

这部分内容将简要介绍如何在 Windows 环境中建立并求解一个小模型。因为模型的结构与操作系统无关，所以如果使用的不是 Windows 系统，仍然需要阅读这一部分内容。但是，值得注意的是，建立模型的方法在不同的操作系统下会略有不同，这将在后面介绍。在 Windows 环境下启动 LINGO 时，将会看到类似图 1-2 所示的 LINGO 初始界面。


**图 1-1　密码输入界面**

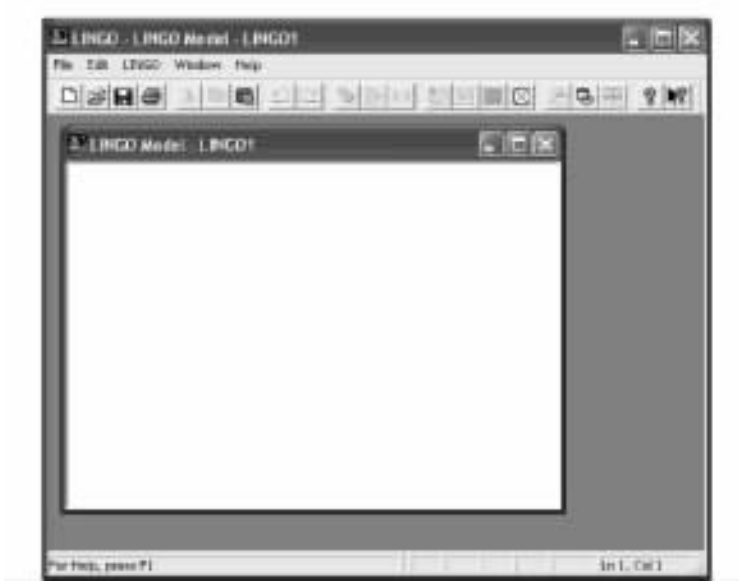


**图 1-2　LINGO 初始界面**

**图 1-2 的最外层窗口是主窗口。主窗口的顶部包括所有的命令菜单和工具栏。主窗口的底部有一个状态栏，它提供 LINGO 当前运行状态的各种即时信息。工具栏和状态栏均可以通过使用 LINGO|Option 命令显示或隐藏。**

**标有 LINGO Model-LINGO1 的子窗口是一个新建的、空白的模型窗口。LINGO 模型都是在这样的子窗口中输入的。**

### 1.1.3 LINGO 模型实例

假设 CompuQuick 公司生产 Standard 和 Turbo 两种类型的电脑，并且每出售一台 Standard 电脑能够获利 100 美元，而出售一台 Turbo 电脑获利 150 美元。Standard 和 Turbo 电脑生产线的生产能力分别为 100 台/天和 120 台/天。CompuQuick 公司所有工人每天所能提供的劳动时间是 160 小时。每生产一台 Standard 电脑需要一个工时，而每生产一台 Turbo 电脑需要两个工时。现在的问题是在不超过生产能力和劳动力限制的条件下，如何安排 Standard 和 Turbo 两种电脑的生产可使 CompuQuick 公司获利最大。

一般说来，一个最优化模型由以下三个部分构成。

1) 目标函数。目标函数是用来表示优化对象的函数式。在 LINGO 中，通常表示为利润最大或成本最低等。一个模型最多只能有一个目标函数。在上述 CompuQuick 公司的实例中，目标函数就是以一定比例生产 Standard 和 Turbo 两种电脑所获得的利润。

2) 变量。变量也称为决策变量，是可以控制的量。最优化的目的就是在满足一切约束条件的前提下确定变量的值，以获得目标函数的最优解。在这个实例中有两个变量，它们分别是 Standard 电脑的生产数量 STANDARD 和 Turbo 电脑的生产数量 TURBO。

3) 约束条件。大多数模型对变量的取值都有一定的限制，至少包括一种资源的限制 (例如时间、原材料、部门预算等)。这些限制由变量构成的一些表达式组成，因为它们约束了变量的取值范围，所以这些表达式就是约束条件。在 CompuQuick 实例中，每一条生产线的生产能力和劳动力的可使用量均有一个约束条件。

现在就这一实例建立目标函数，分别用 STANDARD 和 TURBO 代表 Standard 和 Turbo 两种计算机的产量。CompuQuick 公司的目标是总利润最大，在公式之前加上“MAX = ”表示求最大值。因此，下列目标函数可以写进模型窗口的第一行：

```
MAX = 100 * STANDARD + 150 * TURBO;
```

注意：LINGO 模型中的每一行必须以分号结束，否则，模型将无法求解。接下来，在目标函数的下面输入生产能力和劳动力这两个约束条件。

```
STANDARD < = 100;
TURBO < = 120;
```

注意 在 LINGO 中，可以用两个符号“ < = ”来代替“≤”，或者甚至可以简单地输入“ < ”来代替“≤”。同样，“ > = ”或者“ > ”可以代替“≥”。最后一个约束条件是劳动力数量，由下式表示：

```
STANDARD + 2 * TURBO < = 160;
```

也就是说，工时总数 (STANDARD + 2 * TURBO) 不得大于 ( < = ) 所提供劳动时间的总量 (160)。

将上述表达式输入完毕后，还可以增加一些注释以提高模型的可读性。这样就得到如图 1-3 所示的模型窗口。

根据用户需要，一个表达式可以分成任意多行书写，但是每个表达式必须以分号结束。例如，用一个表达式表示的目标函数可以写为两行。

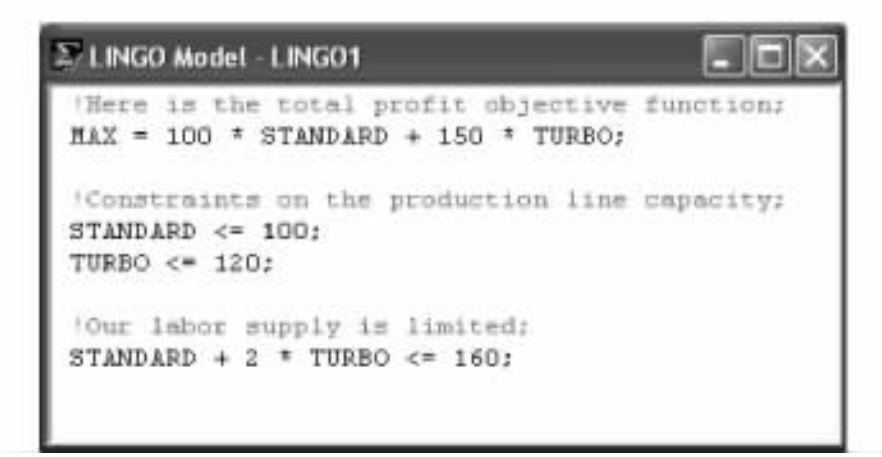



图 1-3　模型窗口

```
MAX = 100 * STANDARD
 + 150 * TURBO;
```

注释部分以感叹号(!)开始,以分号(;)结束。在 LINGO 中,感叹号和分号中间的内容将被忽略而不予执行。注释部分可以占用多行,也可以插入到其他的 LINGO 表达式中。例如:

```
X = 1.5 * Y + Z / 2 * Y; ! This is a comment;
X = 1.5 *  ! This is a comment in the middle of a constraint;
Y + Z / 2 * Y;
```

由于 LINGO 并不区分变量中字母的大小写,所以下列变量名在 LINGO 中是一样的。

```
TURBO
Turbo
turbo
```

但是所有的变量名都必须由字母 A ~ Z 开始,之后可以是字母、数字(0 ~ 9)或者下画线(_),并且变量名的长度限制在 32 个字符内。另外,值得注意的是,LINGO 编辑器能够对语法进行识别,当遇到 LINGO 的关键字时,文本会自动显示为蓝色,注释行则用绿色显示,其他文本用黑色显示。当把光标紧贴一个括号并放到其后时,与之相匹配的另一个括号将会用红色突出显示。这些语法识别功能是非常有用的,可以帮助检查模型中的语法错误。

### 1.1.4　模型的求解

模型建立完成之后就可以求解。选择 LINGO 菜单中的 Solve 命令,或者按下工具栏中的 Solve 按钮(  ),LINGO 就开始对模型进行编译。在这一过程中,LINGO 会进行语法校验,检验是否有语法错误。如果 LINGO 不能通过这一检验,将会出现错误信息提示。例如少用了乘号,将会出现图 1-4 所示的错误信息。

图 1-4 表示这是一个语法错误,并列出了错误的第二行。


图 1-4　错误信息

### 1.1.5　求解状态窗口

如果在编译过程中没用出现语法错误，LINGO 将调用适当的内部算法对模型进行求解。在算法开始执行时，会出现图 1-5 所示的求解状态窗口。


图 1-5　求解状态窗口

求解状态窗口用于监控求解的进度和模型的维数。

在求解状态窗口中有一个中断求解的按钮“Interrupt Solver”。点击这个按钮将使求解过程中断,即导致 LINGO 在下一次迭代前停止算法的运行。在大部分情况下,LINGO 可以回到原来的状态,并且报告到此为止得到的最优解。但是如果在求解不含整型变量的线性模型时被中断,所得解会因为毫无意义而不被采纳。一般情况下,求解线性模型都十分迅速,被中断的可能性非常小。

当算法被中断时所得出的解有如下特点:

1)一定不是最优解;

2)可能不满足所有约束条件;

3)如果模型是线性的,则此结果毫无价值。

点击 Interrupt Solver 按钮旁边的 Close 按钮可关闭求解状态窗口。这个窗口又可通过选择 Windows 菜单中的 Satus Windows 命令辐随时被打开。

在求解状态窗口的左下方有一个 Update interval 文本框。通过自由设置 Update interval 的值,可以控制 LINGO 自动刷新求解状态窗口的时间。若该值被设置为 0,LINGO 将会花费大量的时间来更新求解状态窗口,但是在求解大型模型时,LINGO 并不会以相同的时间间隔来更新求解状态窗口。

#### 1.1.5.1 变量框(Variables)

变量框显示了模型中变量的总数,也显示了非线性变量的数目。如果一个变量存在于模型的非线性约束中,那么它被认为是非线性变量。例如,约束条件 X + Y = 100 是线性的,而 X * Y = 100 是非线性的,则代表变量 X 和 Y 在这个模型中是非线性变量。考察下面的约束式:

```
X * X + Y = 100;
```

式中,X 是非线性的,而 Y 是线性的,不能将 Y 看成是非线性变量。

求解状态窗口的变量框中还给出了模型中整型变量的数目。一般说来,模型中非线性变量和整型变量越多,在一定的时间内求解最优解就越困难。没有整型变量的纯线性模型求解最快。

变量的数目不包括 LINGO 中值已经确定的变量。例如,在约束条件

```
X = 1;
X + Y = 3;
```

中,X 的取值固定为 1,Y 的取值固定为 2,则在模型中 X 和 Y 分别用 1 和 2 代替,不计入变量的总数。

#### 1.1.5.2 约束框(Constraints)

约束框显示的是完全展开的模型中约束条件的总数和非线性约束的总数。如果一个约束条件中有一个以上的非线性变量,则此约束条件就是非线性约束。如果一个约束条件中的所有变量均取定值,则此约束条件为固定约束。固定约束不计入约束总数。

#### 1.1.5.3 非零框 (Nonzeros)

非零框显示的是模型中所有非零系数的总数以及非线性变量的非零系数的总数。在一个约束条件中一般只出现一小部分变量,那些没有出现的变量的隐含系数均为零,而出现的变量的系数都是非零的。因此,所有非零系数的总数是约束条件中所有变量出现的次数。

#### 1.1.5.4 内存使用框 (Generator Memory Used)

内存使用框显示正在使用的模型占用内存的大小,也可以用 LINGO|Opions 命令改变分配内存的大小。

#### 1.1.5.5 运行时间框 (Elapsed Runtime)

运行时间框显示了当前生成和求解模型所耗用的总时间,它可能会受系统中其他运行程序的影响。

#### 1.1.5.6 求解状态框 (Solver Status)

求解状态框显示了算法运行的当前状态。表 1-1 进一步解释了这一部分的各项内容。

表 1-1 求解状态框字段描述

| 字段 | 描述 |
|---|---|
| Model Class (模型类别) | 显示模型的类别,包括 LP、QP、ILP、IQP、PILP、PIQP、NLP、INLP 和 PINLP |
| State (状态) | 显示当前结果的状态,可能的状态包括 Global Optimum (全局最优)、Local Optimum (局部最优)、Feasible (可行)、Infeasible (不可行)、Unbounded (无界)、Interrupted (中断) 和 Undetermined (未确定) |
| Objective (目标值) | 目标函数当前取值 |
| Infeasibility (不可行) | 不能满足的约束的数目 |
| Iterations (迭代) | 算法迭代的次数 |

下面对求解状态框中的各项内容进行详细解释。

1.模型类别字段 (Model Class)

模型类别字段概括了模型的特征,表 1-2 列举了由简单到复杂的所有模型的类别。

表 1-2 模型的类别

| 类别缩写 | 类别 | 描述 |
|---|---|---|
| LP | Linear Program (线性规划) | 所有表达式均为线性的,并且模型中无整型变量 |
| QP | Quadratic Program (二次规划) | 所有表达式都是线性的或二次型的,模型是凸的且无整型变量 |
| ILP | Integer Linear Program (整数线性规划) | 所有表达式均是线性的且部分变量为整型变量 |

续表

| 类别缩写 | 类别 | 描述 |
|---|---|---|
| IQP | Integer Quadratic Program<br>(整数二次规划) | 所有表达式均是线性的或是二次型的,模型是凸的且部分变量为整型变量 |
| PILP | Pure Integer Linear Program<br>(纯整数线性规划) | 所有表达式均为线性且所有变量均为整型变量 |
| PIQP | Pure Integer Linear Program<br>(纯整数二次规划) | 所有表达式均为线性的或二次型的,模型是凸的且所有变量均为整型变量 |
| NLP | Nonlinear Program<br>(非线性规划) | 至少有一个表达式为非线性的 |
| INLP | Integer Nonlinear Program<br>(整数非线性规划) | 至少有一个表达式为非线性的且部分变量为整型变量。一般说来,这类模型除特别小型的模型外,都很难求解 |
| PINLP | Pure Integer Nonlinear Program<br>(纯整数非线性规划) | 至少有一个表达式是非线性的且所有变量为整型变量,这类模型同样很难求解 |

2.状态字段(State)

当 LINGO 刚开始求解模型时,因为算法还没有来得及为模型产生一个解,所以解的初始状态是"Undetermined"(未确定)。一旦算法开始迭代,状态字段将显示为"Infeasible"(不可行)。此时 LINGO 虽然产生了一些试验性的解,但是没有一个能够满足所有约束条件。当算法找到一个可行解的时候,状态窗口将显示为"Feasible"。直到算法不能再找到更好的解才停止运算。状态字段将显示为"Globle Optimum"或"Local Optimum"。如果模型没有非线性约束,那么局部最优解也就是全局最优解;反之,如果模型含有一个或多个非线性约束,那么局部最优解不一定是全局最优解,也许存在一个更好的峰值优于当前所有取得的解,但是算法找不到它。因此,在非线性模型中,LINGO 可能在取得局部最优解的状态下停止运算。实际上,LINGO 尽管存在一个全局最优解,但是由于上述原因,LINGO 可能找不到它。鉴于此,在建立模型时应尽量使用线性约束。

如果一个模型在无界解状态终止,就意味着目标函数的值会无限增大,而在现实世界中,就意味着能够获得无限大的利润,当然这是不可能的,原因可能在于忽略或者遗忘了模型中的某些约束条件。最后,如果在还没有找到最终解的时候中断了 LINGO 的算法,状态字段将显示为"Interrupted"。

3.目标值字段(Objective)

目标值字段给出了在当前解的情况下的目标函数值。如果模型中没有目标函数,这时目标值字段会显示 N/A。

4.不可行字段(Infeasibility)

不可行字段显示模型中所有被违反的约束的总数。值得注意的是,违反变量限制的约束不计算在不可行字段内。因此,由于超出变量界限而产生不可行解时,不可行字段的值可能是零。LINGO 在求解模型时可能会在内部缩放一个模型,这样在不可行域中显示的个数就不再与未缩放的模型相同。要想确定现行解是否为可行的,请参阅前述状态字段部分。

5.迭代字段(Iterations)

迭代字段显示了 LINGO 中算法迭代的次数。LINGO 完成一次基本操作就称之为一次迭代。一次迭代包括两部分动作:第一是要找一个当前值为零的变量,这个变量如果以一个非零

值引入结果中,结果将变得更优;第二是将这个变量引入模型的解中,并不断增大它的取值,直到一个约束将变为不可行或另一个变量的值被"赶"向零。之后,迭代重新开始。一般说来,模型规模越大,求解所需的迭代次数就越多,每次迭代的时间也会越长。

1.1.5.7 附加求解状态框 (Extended Solver Status)

附加求解状态框显示了 LINGO 的一些专门算法的附加信息。这些算法包括:

1) 分支定界算法 (Brand and Bound Solver);

2) 全局算法 (Global Solver);

3) 多启动算法 (Multistart Solver)。

附加求解状态框中的内容只有在这三个特殊算法运行时才会更新。该框中的内容如表 1-3 所示。

表 1-3 附加求解状态框中的字段

| 字段 | 描述 |
|---|---|
| Solver Type (算法类型) | 上面所讲述的三种专门的算法 |
| Best obj (最优目标) | 显示了当前结果的状态,可能的状态包括 Global Optimum (全局最优)、Local Optimum (局部最优)、Feasible (可行)、Infeasible (不可行)、Unbounded (无界)、Interrupted (中断) 和 Undetermined (未确定) |
| Obj Bound (目标界限) | 目标函数当前取值 |
| Steps (步骤) | 不能满足的约束的数目 |
| Active (次数) | 算法迭代的次数 |

下面对附加求解状态框中的各项内容进行详细解释。

1. 算法类型字段 (Solver Type)

算法类型字段显示了"B-and-B"、"Global"以及"Multistart"等专门算法中的一个。

LINGO 采用分支定界法求解整型变量的模型。分支定界法是一个系统的方法,可以隐含地将整型变量的各种组合一一列举出来。Hiuier 和 Lieberman (1995) 已对分支定界法的算法进行了详细的研究。全局算法和多启动算法是两种专门用于求解非线性模型的算法。许多非线性模型是非凸或者非平滑的。那些依赖于局部搜索过程的非线性算法 (如 LINGO 默认的非线性算法) 对这类模型的作用不大,它们往往收敛于一个局部的、次优的解,而这一解可能与真正的全局最优解相差甚远。全局算法和多启动算法就是针对这种情况而设立的特殊算法。

2. 最优目标和目标界限字段 (Best Obj 和 Obj Bound)

最优目标字段显示当前找到的可行的最佳目标值,而目标界限字段则显示目标函数的边界。这一边界能够给出改善目标函数的程度,所得的最佳目标函数值不会超过边界值。在很多时候,最优目标和目标界限值十分接近,这表明 LINGO 求得的当前最佳解就是最优解或接近最优解。此时,用户可以选择中断算法而节省计算时间。

3. 步骤字段 (Steps)

步骤字段显示的内容与正在运行的专门算法有关。它的含义如表 1-4 所示。

表 1-4　步骤字段的含义

| 求解方法 | Steps 字段解释 |
|---|---|
| 分支定界法 | 分支定界树中分支的数目 |
| 全局法 | 产生的子问题框数目 |
| 多启动法 | 启动的算法数目 |

### 1.1.6　模型结果报告

当 LINGO 求解完 CompuQuick 公司的模型后，会出现一个标有 Solution Report 的窗口，如图 1-6 所示。这个窗口对模型求解的结果进行了详细的描述。

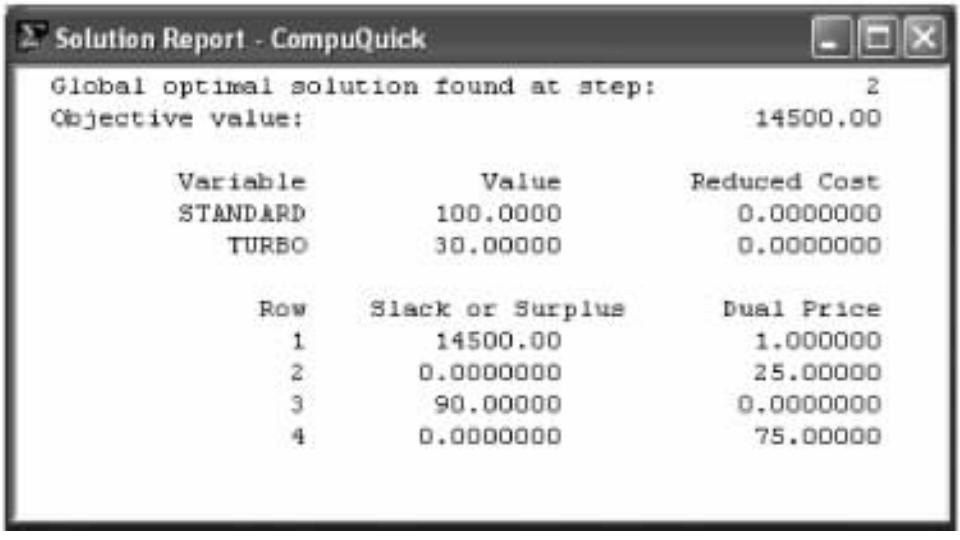



图 1-6　CompuQuick 公司模型求解报告

### 1.1.7　保存

可以使用 File | Save 命令对模型进行保存，或直接点击 图标进行保存。若没有特殊要求，LINGO 将自动为文件加上 .LG4 的扩展名。

## 1.2　在其他操作系统中使用 LINGO

### 1.2.1　命令行提示

在 Windows 以外的操作系统中，要以命令行命令运行 LINGO，所有对 LINGO 的指令都需要采用文本形式的命令行给出。

当启动 LINGO 中的命令行命令时，就会出现以冒号开始的命令提示符，如下所示：

```
LINGO
```

:

冒号提示符代表 LINGO 正在等待命令的输入,问号提示符则说明已经开始执行一个命令并且 LINGO 正等待提供关于此命令的额外信息,比如一个数字或一个名字。如果想退出已经开始的命令,可以在问号提示符后输入一个空白行,这样 LINGO 就返回至冒号提示阶段。

### 1.2.2 在命令行中输入模型

在命令行界面中输入模型前,通过在冒号后输入"MODEL:"以告诉 LINGO 准备输入模型语句。这时 LINGO 会显示一个问号提示符,然后就可以逐行输入模型。

以 CompuQuick 公司的模型为例,在输入完模型之后,屏幕显示如下内容(输入部分用粗体表示):

```
LINGO
:MODEL:
?MAX = 100 * STANDARD + 150 * TURBO;
?STANDARD < = 100;
?TURBO < = 120;
?STANDARD + 2 * TURBO < = 160;
?END
:
```

END 表示结束模型的输入。在输入 END 命令后,随之就会显示冒号提示符,表明模型已被写入内存并准备求解。

### 1.2.3 在命令行中求解模型

在命令行中输入 GO 命令后按回车键,LINGO 开始编译模型。这也是 LINGO 检查语法错误的过程。若不能通过检验,会出现错误信息提示。

如果在编译过程中没有发现语法错误,LINGO 会启动适当的内部算法寻找模型的最优解。以 CompuQuick 公司的模型为例,当 LINGO 求解完模型时,会出现如下求解报告:

```
Optimal solution found at step:2
Objective value:14500.00
Variable           Value          Reduced Cost
STANDARD        100.0000           0.0000000
TURBO           30.00000           0.0000000

Row      Slack or Surplus          Dual Price
  1             14500.00            1.000000
  2            0.0000000            25.00000
  3             90.00000           0.0000000
  4            0.0000000            75.00000
```

### 1.2.4 在命令行中打印和保存

DIVERT 文件命令用来将所有的 LINGO 报告发送到一个文件里，而不是屏幕上。之后，就可以将这个文件发送至打印机或将它载入一个用于处理文字的程序进行打印。

例如，想通过新建一个包含模型和求解结果副本的文本文件以实现打印，需输入下列命令：

```
DIVERT MYFILE        !打开名为 MYFILE 的输出文件；
LOOK ALL             !输出模型到 MYFILE；
GO                   !输出求解结果到 MYFILE；
RVRT                 !关闭输出文件；
```

若想将模型存入磁盘，需使用 SAVE 命令，并且命令后紧接文件名。例如，命令

```
SAVE MYFILE.LNG
```

用于保存当前模型的副本到名为 MYFILE.LNG 的文件中。在以后的使用过程中，可用 TAKE 命令调出已保存的模型。

## 1.3 研究模型的结果报告

### 1.3.1 介绍

CompuQuick 实例中，结果报告显示如下内容：

1）LINGO 经过两次迭代求得模型最优解；

2）获得的最大利润为 $ 14 500；

3）需要生产的 Standard、Turbo 两种计算机的数量分别为 100 台和 30 台。

值得注意的是，由于 Turbo 类型的计算机需要更加密集的劳动力，所以这种利润更高的机型反倒要少生产。

### 1.3.2 Reduced Cost（降低的成本）

在 LINGO 的结果报告中，每一个变量都对应一个 Reduced Cost 值。关于这个值有两种等价的解释。

第一种解释：在最优化问题中，要想给问题中的变量一个确定的值，使其进入基解，则必须改变该变量在目标函数中的系数。例如，在一个最大（小）化问题中，如果一个变量有 10 个单位的 Reduced Cost 值，为了使这个变量进入基解，它在目标函数中的系数就必须增加（减少）10 个单位。最优解中的变量的 Reduced Cost 值自动取零，例如上例中的 STANDARD 和 TURBO。

第二种解释：变量在解中的数值增加一个单位，目标函数就必须相应地增加或减少若干单位。如果一个变量的 Reduced Cost 值是 10 个单位，为了使该变量在解中增加一个单位，则最大（小）化问题的目标函数值就会增加（减少）10 个单位。

Reduced Cost 值只在一定的取值范围内有效。关于决定 Reduced Cost 取值范围的更详细的

信息请参阅第 5 章中的 LINGO|Range 命令。

### 1.3.3 Slack (松弛变量值) or Surplus (剩余变量值)

LINGO 结果报告中的 Slack or Surplus 列说明了约束条件距离成为一个等式存在的差距。在"< = "的约束中,差值称为 Slack (松弛变量值);在"> = "的约束中,差值则称为 Surplus (剩余变量值)。如果一个约束条件成为等式后,Slack 或 Surplus 的值就为零。如果一个约束条件不能满足,即无可行解,Slack 或 Surplus 的值就为负数。了解这些情况可以帮助用户找到不可行模型 (没有解满足所有约束条件) 的矛盾约束。松弛的约束 (这些约束中的松弛或剩余变量值大于零) 在这一栏的值为正数。

在 CompuQuick 公司的模型求解报告中,注意到第三行 (TURBO < = 120) 的 Slack 值为 90。这是由于 TURBO 的最优值是 30,要达到 120 还差 90。

### 1.3.4 Dual Price (对偶变量值)

LINGO 的求解报告中还有一个 Dual Price 值。这个值可以解释为:当约束条件右边的常数增加一个单位时,目标函数发生相应数值的变化。例如,在 CompuQuick 实例中,第四行的对偶变量值为 75,这意味着增加一个劳动力可使目标函数值增加 75,达到 14 575。注意:在最大化问题中目标函数值是增加,而在最小化问题中即使增加约束条件右边的值,目标函数值也是减小。

Dual Price 有时也称为影子价格,这是因为它代表购买附加的单位资源的期望价格。在 CompuQuick 实例中表示 CompuQuick 公司最多愿意用 75 美元购买一个劳动力。和 Reduced Cost 一样,Dual Price 也是在一定范围内有效。关于决定 Dual Price 的取值范围的详细信息请参阅第 5 章中的 LINGO|Range 命令。

## 1.4 LINGO 模型语言介绍

在了解 LINGO 在各种操作系统中的使用方法之后,下面对 LINGO 模型语言进行简单的介绍,使用户掌握简单的语言规则和使用技巧。关于 LINGO 模型语言详细的介绍请参见以后的章节。

### 1.4.1 LINGO 模型语言的特点

LINGO 最强大的功能之一就是它的数学模型语言,LINGO 模型语言与标准的数学符号十分相似,用它描述问题特别自然。例如,可以用一个紧凑的表达式表示一系列相似的约束条件,而不是将每一个约束条件单独表示,这样可使模型得到更好的维护和扩展。

LINGO 模型语言的另一个便捷功能是数据功能。数据功能使模型的数据和相应的表达式分离。实际上,LINGO 可以从电子表格、数据库和文本文件中导入数据,利用独立于模型的数据会使调试模型非常容易,而且可以减少出现错误的机会。

前面讨论的 CompuQucik 模型使用的是标量变量法,也就是把每一个变量名称 (例如 STANDARD 和 TURBO) 清楚地列出来,每一个约束条件 (例如 TURBO < = 120) 也明确地表示出

来。在一些大型的模型中，有可能会遇到很多十分类似的约束条件和变量，如果使用标量变量法就必须将每个变量、约束条件逐个地输入，这样要做大量重复性的工作。幸运的是 LINGO 具有集合功能，可以简单有效地完成这样的操作。

## 1.4.2 运输模型

以下是一个关于运输模型的实例，用来说明怎样利用集合功能。

### 1.4.2.1 问题简述

假设 Wireless Widget (WW) 公司有 6 个仓库向 8 个销售商供应小装饰品，每个仓库的供应量是有限的，而每个销售商的需求又必须得到满足。因此，WW 公司要决定每个仓库出多少货能满足每个销售商而运输成本又最低。这个传统的最优化问题是十分典型的运输问题 (图 1-7)。

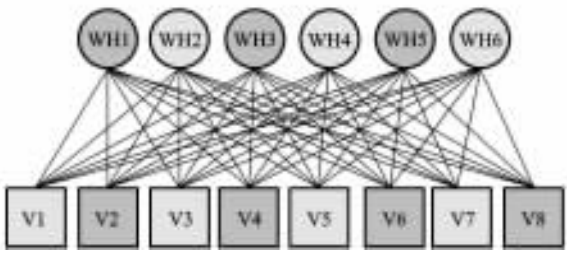



图 1-7 运输问题示意图

因为每个仓库都可以出货给每个销售商，所以共有 48 条运输路线，故需要对每条路线定义一个变量来表示这条路线的运输量。

### 1.4.2.2 数据

本运输模型用到的相关数据如表 1-5、1-6、1-7 所示。

表 1-5 存货数据

| 仓库号 (WH) | 货物量 |
|---|---|
| 1 | 60 |
| 2 | 55 |
| 3 | 51 |
| 4 | 43 |
| 5 | 41 |
| 6 | 52 |

表 1-6 销售商需求

| 销售商号 (V) | 销售商需求量 |
|---|---|
| 1 | 35 |
| 2 | 37 |
| 3 | 22 |
| 4 | 32 |
| 5 | 41 |
| 6 | 32 |
| 7 | 43 |
| 8 | 38 |

表 1-7　运输成本（$）

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 |
|---|---|---|---|---|---|---|---|---|
| WH1 | 6 | 2 | 6 | 7 | 4 | 2 | 5 | 9 |
| WH2 | 4 | 9 | 5 | 3 | 8 | 5 | 8 | 2 |
| WH3 | 5 | 2 | 1 | 9 | 7 | 4 | 3 | 3 |
| WH4 | 7 | 6 | 7 | 3 | 9 | 2 | 7 | 1 |
| WH5 | 2 | 3 | 9 | 5 | 7 | 2 | 6 | 5 |
| WH6 | 5 | 5 | 2 | 2 | 8 | 1 | 4 | 3 |

### 1.4.2.3　目标函数

建立模型的第一步是建立目标函数，WW 公司的目标是运输成本最低。如果用 VOLUME_I_J 表示从 I 仓库到 J 销售商运送的商品数量，用标量变量书写目标函数，就得到如下形式：

```
MIN = 6 * VOLUME_1_1 + 2 * VOLUME_1_2 +
6 * VOLUME_1_3 + 7 * VOLUME_1_4 +
4 * VOLUME_1_5 +
⋮
8 * VOLUME_6_5 + VOLUME_6_6 + 4 * VOLUME_6_7 +
3 * VOLUME_6_8;
```

现实生活中更多的情况是销售商数以千计，用标量变量法很容易出错，所以，可以用大家都很熟悉的数学符号表示冗长的表达式。例如：

$$\text{Minimize} \sum_{ij} \text{COST}_{ij} \cdot \text{VOLUME}_{ij}$$

与上式类似，LINGO 的模型语言可以用下列简单而容易理解的表达式表示目标函数：

```
MIN = @SUM(LINKS(I,J): COST(I,J) * VOLUME(I,J));
```

上式表示将所有路线（LINK）上的运输货物（VOLUME）成本（COST）的和最小化。表 1-8 对目标函数中的数学符号和 LINGO 模型语言进行比较。

表 1-8　数学符号与 LINGO 语法

| 数学符号 | LINGO 语法 |
|---|---|
| Minimize | MIN = |
| $\sum_{ij}$ | @SUM(LINKS(I,J):) |
| $\text{COST}_{ij}$ | COST(I,J) |
| · | * |
| $\text{VOLUME}_{ij}$ | VOLUME(I,J) |

#### 1.4.2.4 约束条件

在建立目标函数之后,下一步是用表达式表示约束条件。在此模型中有两类约束条件:第一类用于表示需求约束,即满足每个销售商的供货需求;另一类为能力约束,表示保证每个仓库的货物运出量不会超过其供货能力。

需求约束可以用数学符号通过下式表示:

$$\sum i \ VOLUME_{ij} = DEMAND_j, \text{ for all } j \text{ in } VENDORS$$

相应的 LINGO 模型语言如下:

```
@FOR(VENDORS(J):
  @SUM(WAREHOUSES(I):VOLUME(I,J)) =
    DEMAND(J));
```

上面的 LINGO 表达式覆盖了 8 个需求约束,表示对所有销售商来说,各个仓库出货量之和等于销售商的需求量。表 1-9 对需求约束中的数学符号和 LINGO 模型语言进行比较。

表 1-9 需求约束 LINGO 语法

| 数学符号 | LINGO 语法 |
|---|---|
| for all j in VENDORS | @FOR(VENDORS(j): |
| $\sum i$ | @SUM(WAREHOUSES(i): |
| VOLUME ij | VOLUME(I,J)) |
| = | = |
| DEMAND j | DEMAND(j)); |

同样,能力约束用标准的数学符号表示为:

$$\sum j \ VOLUME\ ij <= CAPACITY\ i, \text{ for all } i \text{ in } WAREHOUSES$$

与其相应的 LINGO 模型语言表达如下:

```
@FOR(WAREHOUSES(I):
@SUM(VENDORS(J):VOLUME(I,J)) <=
  CAPACITY(I));
```

对于每个仓库来说,销售商的需求量之和一定小于等于该仓库的总存货量。

#### 1.4.2.5 完整模型

将上面讨论过的内容结合起来就构成下面完整的运输模型:

```
MODEL:
MIN = @SUM(LINKS(I,J):
    COST(I,J) * VOLUME(I,J));
@FOR(VENDORS(J):
```

```
    @SUM(WAREHOUSES(I):VOLUME(I,J)) =
     DEMAND(J));
  @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J):VOLUME(I,J)) < =
     CAPACITY(I));
END
```

**但是,用户还需要在上述 LINGO 模型的集合部分和数据部分定义数据。**

### 1.4.2.6 定义集合

**在实践中建立模型时,会发现存在一个或多个包含相关对象的集合,例如工厂、消费者、运输车辆和雇员均为集合。通常,如果一个约束条件针对某个集合中的一员,则往往适用于集合中的其他成员。这一简单的概念就是 LINGO 模型语言的核心。LINGO 允许在集合域中定义相关对象的集合。集合域以"SETS:"开始,并且自占一行;以"ENDSETS"结束,同样也自占一行。定义集合成员后,就可以用 LINGO 的集合循环函数(例如 @FOR)对集合所有成员进行操作。第 2 章将介绍更多相关概念。**

**在运输模型实例中,建立了下面三个集合:**

**1)仓库;**

**2)销售商;**

**3)从仓库到销售商的运输路线。**

**定义这三个集合的模型集合域如下:**

```
SETS:
   WAREHOUSES/WH1 WH2 WH3 WH4 WH5 WH6/:CAPACITY;
   VENDORS/V1 V2 V3 V4 V5 V6 V7 V8/    :DEMAND;
   LINKS(WAREHOUSES, VENDORS):COST, VOLUME;
ENDSETS
```

**第二行语句说明 WAREHOUSES 集合有 WH1、WH2、……、WH6 六个元素,每个元素有一个属性为 CAPACITY。第三行是八个销售商的集合,每个元素有一个属性为 DEMAND。**

**最后一个集合称为 LINKS,代表 48 条运输路线。每条路线都有单位运价 COST 和运输量 VOLUME 两个属性。定义这个集合使用的语法不同于前面的两个集合。LINKS 集合是由 WAREHOUSES 和 VENDORS 两个集合派生出来的,其 48 个元素分别取自两个集合的元素组成的所有序对。为了清楚起见,表 1-10 列出 LINKS 集合示意元素。**

**表 1-10 LINKS 集合示意元素**

| 元素索引 | 运输路线 |
|---|---|
| 1 | WH1→V1 |
| 2 | WH1→V2 |
| 3 | WH1→V3 |
| ⋮ | ⋮ |
| 47 | WH6→V7 |
| 48 | WH6→V8 |

#### 1.4.2.7 输入数据

LINGO 用到的数据可以放在模型的数据域中。在运输模型实例中,数据域如下所示:

```
DATA:
   CAPACITY = 60 55 51 43 41 52;
   DEMAND = 35 37 22 32 41 32 43 38;
   COST = 6 2 6 7 4 2 5 9
          4 9 5 3 8 5 8 2
          5 2 1 9 7 4 3 3
          7 6 7 3 9 2 7 1
          2 3 9 5 7 2 6 5
          5 5 2 2 8 1 4 3;
ENDDATA
```

与集合域一样,数据域以"DATA:"开始,自占一行;以"ENDDATA"结束,也自占一行。WAREHOUSES 集合的 CAPACITY 属性和 VENDORS 集合的 DEMAND 属性的赋值方式较为直接,而对二维集合 LINKS 的 COST 属性进行赋值相对比较复杂。当 LINGO 在数据域中对一个多维数列进行初始化时,首先对处于外层的集合按递增的原则进行索引。在此例中,首先对 COST(WH1,V1)进行初始化,之后是 COST(WH1,V2)至 COST(WH1,V8),然后是 COST(WH2,V1),依此类推。

在此例中,数据直接放在模型的数据域中。此外,LINGO 还可以从外部数据源引入数据,例如文本文件、电子表格和数据库等。建立 OLE 与 Excel 连接,或创建 ODBC 连接一些流行的数据库均可达到上述目的。这样,数据较为独立,非常容易修改并且减少了出错的可能,对求解一个数据经常发生变化的模型来说是非常有用的。

将数据域、集合域、目标函数和约束条件放在一起就形成了完整的模型,并且添加了一些注解以增强模型的可读性。本例完整的模型表示为下列形式:

```
MODEL:
!运输问题;
!集合域;
SETS:
   WAREHOUSES/ WH1 WH2 WH3 WH4 WH5 WH6/:CAPACITY;
   VENDORS/ V1 V2 V3 V4 V5 V6 V7 V8/:DEMAND;
   LINKS (WAREHOUSES, VENDORS):COST, VOLUME;
ENDSETS
!目标函数;
MIN = @SUM(LINKS(I,J):
    COST(I,J) * VOLUME(I,J));
!需求约束;
@FOR(VENDORS(J):
    @SUM(WAREHOUSES(I):VOLUME(I,J)) =
```

```
        DEMAND(J));
    !能力约束;
    @FOR(WAREHOUSES(I):
        @SUM(VENDORS(J):VOLUME(I,J)) <=
        CAPACITY(I));
    !数据域;
    DATA:
        CAPACITY = 60 55 51 43 41 52;
        DEMAND = 35 37 22 32 41 32 43 38;
        COST = 6 2 6 7 4 2 5 9
               4 9 5 3 8 5 8 2
               5 2 1 9 7 4 3 3
               7 6 7 3 9 2 7 1
               2 3 9 5 7 2 6 5
               5 5 2 2 8 1 4 3
    ENDDATA
    END
```

#### 1.4.2.8 求解运输模型

建立完模型后就可以求解,以找到从每个仓库运输货物的最优方案。如前所述,在 Windows 操作系统下的 LINGO 中,可选择 LINGO 菜单中的 Solve 命令或直接点击 按钮求解模型 在其他操作系统下,例如在命令行提示版本中输入 GO 命令即可,然后 LINGO 就会求解模型并输出一个可能很长的结果报告,包括所有变量的值、约束条件和模型中的数据。

如果不需要 LINGO 输出完整的结果报告,在 Windows 操作系统中,选择 LINGO 菜单中 Options 命令下的 Interface 页中的 Terse output 即可。这样 LINGO 在结果窗口中就只显示迭代次数和模型的目标值。在非 Windows 版本中,输入 TERSE 命令后再输入 GO 命令,同样可以实现此功能。

为了获得一个仅含有非零的 VOLUME 的报告,可以选择 LINGO 菜单中的 Solution 命令,就会出现图 1-8 所示的对话框。

在 Attribute or Row Name 字段中点击下拉按钮选择 VOLUME,并选中 Nonzeros Only 复选框。这样,单击 OK 按钮后,就会得到图 1-9 所示的结果报告。

如果在非 Windows 操作系统下运行 LINGO,输入 Nonzero VOLUME 命令可以得到同样的结果报告。

#### 1.4.2.9 总结

总而言之,LINGO 有两种类型的集合,即基本集合和衍生集合。基本集合的元素代表模型中的基本对象,不能被分解,它可以用显式列表或隐式列表进行定义。当使用显式列表时,必须逐个地输入集合中的每个元素 ;使用隐式列表时,只要输入第一个和最后一个集合元素就可以,LINGO 会自动生成中间的元素。衍生集合则是由其他的成员集合构成,这些成员集合可看

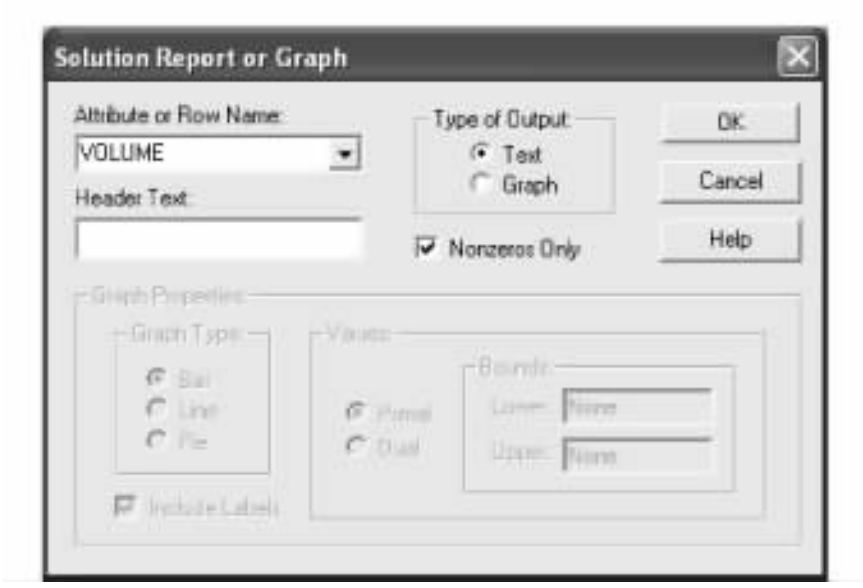
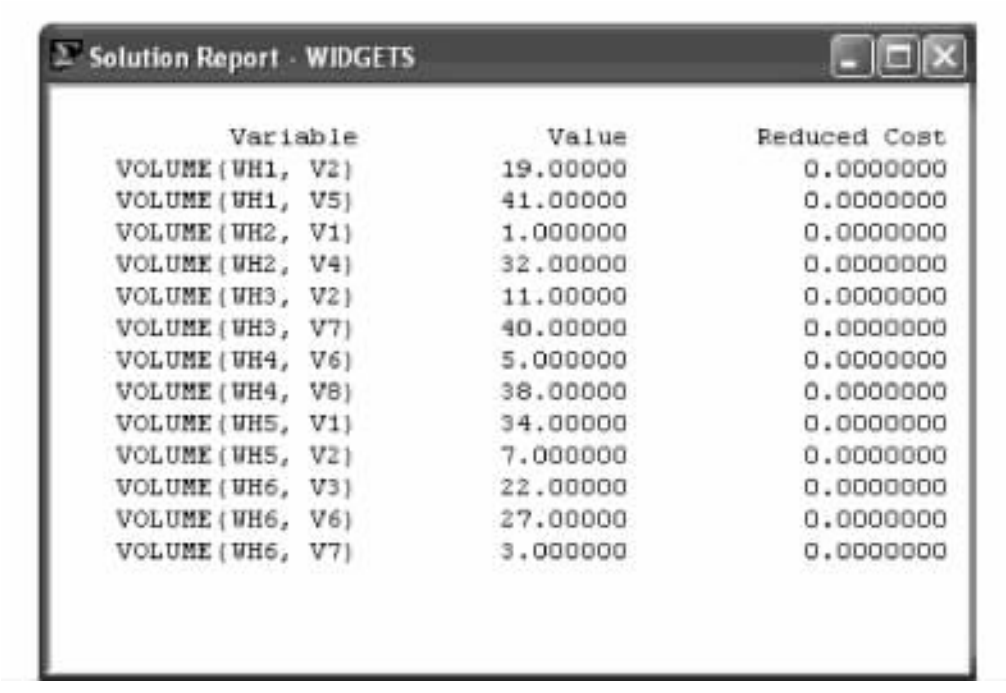



**图 1-8　求解报告或图表对话框**




| Variable | Value | Reduced Cost |
|---|---|---|
| VOLUME(WH1, V2) | 19.00000 | 0.0000000 |
| VOLUME(WH1, V5) | 41.00000 | 0.0000000 |
| VOLUME(WH2, V1) | 1.000000 | 0.0000000 |
| VOLUME(WH2, V4) | 32.00000 | 0.0000000 |
| VOLUME(WH3, V2) | 11.00000 | 0.0000000 |
| VOLUME(WH3, V7) | 40.00000 | 0.0000000 |
| VOLUME(WH4, V6) | 5.000000 | 0.0000000 |
| VOLUME(WH4, V8) | 38.00000 | 0.0000000 |
| VOLUME(WH5, V1) | 34.00000 | 0.0000000 |
| VOLUME(WH5, V2) | 7.000000 | 0.0000000 |
| VOLUME(WH6, V3) | 22.00000 | 0.0000000 |
| VOLUME(WH6, V6) | 27.00000 | 0.0000000 |
| VOLUME(WH6, V7) | 3.000000 | 0.0000000 |

**图 1-9　模型结果报告**

**作是衍生集合的父集合，成员集合可以是衍生集合也可以是基本集合。衍生集合又可分为稠密集合和稀疏集合。稠密集合中含有父集合元素的全部组合（有时称为笛卡儿乘积或父集合的交叉）。稀疏集合是稠密集合的一个子集，可直接列出其元素或通过元素过滤器产生。元素过滤方法就是通过判断集合元素是否满足某个逻辑条件来清楚、简洁地列出稀疏集合的所有元素。**

## 1.5 LINGO 语言的其他特征

### 1.5.1 约束条件的命名

LINGO 允许在模型中对约束条件进行命名。这样有两个好处：首先，使用了约束名的结果报告便于阅读；其次，LINGO 的很多错误信息都通过名字指向约束。如果不对约束命名，要找出产生错误的源头将很困难。

LINGO 并没有要求一定要对约束条件命名，如果不对其命名，LINGO 将会用内部索引自动对约束命名。这种内部索引与定义约束条件的顺序不大相同，这就使得对结果报告和错误信息的理解变得比较困难。因此，最好在建立模型时对所有的约束进行命名。

命名约束条件十分简单，只要在一个约束开始处插入一个用方括号括起来的名字即可。值得注意的是，名字必须以字母 A ~ Z 开头，后面可以是任意字符，包括字母、数字（0 ~ 9）或下画线（_），长度不能超过 32 个字符。下面举例说明。

命名模型的目标为 OBJECTIVE：

```
[OBJECTIVE] MIN = X;
```

将运输模型的需求约束命名为 DEMAND_ROW：

```
@FOR(LINKS(I,J):[DEMAND_ROW]
        @SUM(SOURCES(I):SHIP(I,J)) > = DEMAND(J));
```

为了进一步解释这些名称，这里特地更改了 WIDGETS 模型，增加了粗体的约束名称，如下所示：

```
MODEL:
!运输问题;
!集合域;
SETS:
   WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/ :CAPACITY;
   VENDORS / V1 V2 V3 V4 V5 V6 V7 V8/    :DEMAND;
   LINKS(WAREHOUSES,VENDORS) :COST,VOLUME;
ENDSETS
!目标函数;
   [OBJECTIVE] MIN = @SUM(LINKS(I,J):
    COST(I,J) * VOLUME(I,J));
!需求约束;
       @FOR(VENDORS(J):[DEMAND_ROW]
    @SUM(WAREHOUSES(I):VOLUME(I,J)) =
     DEMAND(J));
!能力约束;
       @FOR(WAREHOUSES(I):[CAPACITY_ROW]
```

```
    @SUM(VENDORS(J):VOLUME(I,J)) <=
     CAPACITY(I));
!数据域;
DATA:
   CAPACITY=60 55 51 43 41 52;
   DEMAND=35 37 22 32 41 32 43 38;
   COST=6 2 6 7 4 2 5 9
        4 9 5 3 8 5 8 2
        5 2 1 9 7 4 3 3
        7 6 7 3 9 2 7 1
        2 3 9 5 7 2 6 5
        5 5 2 2 8 1 4 3;
ENDDATA
END
```

**模型求解后得到下列结果报告就很容易理解了。**

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| OBJECTIVE | 664.0000 | 1.000000 |
| DEMAND_ROW(V1) | 0.0000000 | -4.000000 |
| DEMAND_ROW(V2) | 0.0000000 | -5.000000 |
| DEMAND_ROW(V3) | 0.0000000 | -4.000000 |
| DEMAND_ROW(V4) | 0.0000000 | -3.000000 |
| DEMAND_ROW(V5) | 0.0000000 | -7.000000 |
| DEMAND_ROW(V6) | 0.0000000 | -3.000000 |
| DEMAND_ROW(V7) | 0.0000000 | -6.000000 |
| DEMAND_ROW(V8) | 0.0000000 | -2.000000 |
| CAPACITY_ROW(WH1) | 0.0000000 | 3.000000 |
| CAPACITY_ROW(WH2) | 22.00000 | 0.000000 |
| CAPACITY_ROW(WH3) | 0.0000000 | 3.000000 |
| CAPACITY_ROW(WH4) | 0.0000000 | 1.000000 |
| CAPACITY_ROW(WH5) | 0.0000000 | 2.000000 |
| CAPACITY_ROW(WH6) | 0.0000000 | 2.000000 |

**要注意每一行的前面都不是简单的索引数,而是一个易于理解的名字。如果用 @FOR 函数将约束条件建立在一个集合之上,则 LINGO 对集合中的每个元素给出约束名称,并在紧跟其后的圆括号内给出相应的元素。**

## 1.5.2 模型的标题

**如同输入一个约束条件一样,可以在任何地方输入一个模型的标题,并在模型求解后出现在结果报告的顶部。**

**模型的标题必须由关键字"TITLE"开始,用一个分号结束。TITLE 和分号之间的所有文本都是模型的标题。接下来在 WIDGETS 开始处添加一个标题:**

```
MODEL:
TITLE Widgets;
!A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
   WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/:CAPACITY;
⋮
```

**注意这时的结果报告顶部含有模型的标题：**

```
Model Title:WIDGETS
     Variable          Value          Reduced Cost
CAPACITY(WH1)        60.00000          0.0000000
CAPACITY(WH2)        55.00000          0.0000000
CAPACITY(WH3)        51.00000          0.0000000
CAPACITY(WH4)        43.00000          0.0000000
                     ⋮
```

## 1.6 问题的最大规模

LINGO 的一些版本限制了模型的变量、整型变量、非线性变量和约束条件的数目。变量总数的限制用于限制模型中可最优化的变量的总体数量。整型变量数目限制用来确定采用@BIN或@GIN 函数定义的并且可最优化的整型变量的总数。非线性变量数目用于限制非线性约束中可最优化变量的总数。

LINGO 根据现有的版本决定能够解决的最优化问题的最大规模，如表 1-11 所示。

表 1-11　LINGO 不同版本的限制

| 版本 | 变量总数 | 整型变量总数 | 非线性变量总数 | 约束条件总数 |
|---|---|---|---|---|
| Demo/Web | 300 | 30 | 30 | 150 |
| SolverSuite | 500 | 50 | 50 | 250 |
| Super | 2 000 | 200 | 200 | 1 000 |
| Hyper | 8 000 | 800 | 800 | 4 000 |
| Industrial | 32 000 | 3 200 | 3 200 | 16 000 |
| Extended | 无限 | 无限 | 无限 | 无限 |

用户可以从 Help 菜单中选择 About LINGO 命令了解所安装版本的求解能力。如果需要一个大容量的 LINGO 版本，可以在 LINDO 公司的网站上升级原有的版本，并且联系 LINGO 供应商询问价钱和送货方式。

# 第 2 章　集合的使用

如前所述,在建立模型时通常会遇到一组或多组相关的对象,比如工厂、顾客、交通工具或雇员等。LINGO 允许把这些相关的对象组织在一起并放在集合中。一旦模型中的对象组成集合,就可以通过函数利用各种集合,实现 LINGO 模型语言的全部功能。

## 2.1　使用集合的目的

集合是 LINGO 建模语言的基础,是 LINGO 程序中功能最强大的基础模块。掌握了集合的使用方法,就能用一个简单的语句写出一系列相似的约束条件,也可以用简短的语句表达冗长复杂的表达式,这样就可以迅速而容易地建立一个大型模型。

在大型模型中,通常会遇到需要表达一组相似的计算和约束的情况,而 LINGO 处理数据集合的能力,能够高效地实现这种操作。

例如,一个 100 个仓库的运输模型,要明确地表达每一个约束一定是非常冗长乏味的(例如,仓库 1 的运货量不能超过它的存货量,仓库 2 的运货量不能超过它的存货量,仓库 3 的运货量不能超过它的存货量,等等)。LINGO 提供了表达这些不等式的最简单的形式,方法容易理解和掌握,这就是集合的作用。

## 2.2　集合简介

集合是一些相关对象的全体。集合中的每一个元素都有一个或多个与之相关的特征,这些特征称为属性。

在求解 LINGO 模型时,这些属性可以已知或未知。例如,在一个产品集合中,可以将每一个产品的价格作为属性;在一个卡车的模型中,可以将每一辆卡车的运输能力作为属性;在一个雇员模型中,可以将每一个雇员的工资作为属性,也可以将每一个雇员的生日作为属性。

LINGO 具有基本集合和衍生集合两种类型。

基本集合的元素不能被进一步简化。在 Wireless Widgets 公司的例子中,集合 WAREHOUSES 中包含 6 个仓库,是一个基本集合。同样,由 8 个销售商组成的集合也是一基本集合。

衍生集合可以被定义为使用一个或多个其他集合的集合。换句话说,衍生集合是由已存在的集合作为元素衍生形成的。在 Wireless Widgets 公司的例子中,由 6 个仓库和 8 个销售商之间的运输路线组成的 LINKS 集合就是一个衍生集合。这一集合中的元素是由 WAREHOUSES 集合和 VENDORS 集合中的元素一一配对形成。这里,LINKS 集合是由两个基本集合唯一形成的。此外,衍生集合也可以由其他的衍生集合形成。

## 2.3 模型的集合域

集合可以在 LINGO 模型的任意位置定义,定义集合的语句被称为集合域。集合域以关键字"SETS:"开始,以关键字"ENDSETS"结束。一个模型中可以没有集合域,也可以有一个或多个集合域,并且集合域可以出现在模型的任意位置,唯一的限制就是必须在模型的约束条件使用集合之前定义集合及其属性。

### 2.3.1 基本集合的定义

在集合域中定义一个基本集合时,要包括如下参数:

1) 集合的名称;

2) 集合包含的元素,即集合中的对象 (可选);

3) 集合中元素的所有属性 (可选)。

基本集合在定义时采用以下语法 (使用方括号表示此项是可选的):

```
setname [/ member_list /] [: attribute_list];
```

集合的名称 (setname) 是用来标志集合的,它最好具有一定的描述性,以便记忆或区分。命名集合有一定的规则:集合名称必须以字母开头,后面是其他字母或下画线 (_)。LINGO 集合名称中的字母不区分大小写。

元素列表 (member_list) 是组成集合的元素的列表。如果集合元素包含在集合的定义中,可以将它们直接或间接列出;如果集合元素不包含在集合的定义中,可以随后在模型的数据域中对其进行定义。

要把集合元素直接列出,需要赋给每个集合元素一个唯一的名称,还可以选择用逗号把它们彼此隔开。与集合的名称一样,元素的名称也必须遵守命名规则。例如,在运输模型中,使用直接元素列表定义集合 WAREHOUSES:

```
WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/ : CAPACITY;
```

当使用间接集合元素列表时,就不必把每一个集合元素的名称列出来。利用间接集合元素列表的语法如下:

```
setname / member1..memberN / [: attribute_list];
```

member1 是集合中的第一个元素,memberN 是集合的最后一个元素。LINGO 自动生成 member1 到 memberN 之间的元素名称。这样就非常简洁、方便地建立了一个基本集合。有一点必须注意,起始元素名称和末端元素名称的格式必须一致。表 2-1 包含所有可能用到的间接集合元素列表的格式。

表 2-1 间接集合元素列表的格式

| 格式 | 举例 | 集合元素 |
|---|---|---|
| 1..n | 1..5 | 1,2,3,4,5 |

续表

| 格式 | 举例 | 集合元素 |
|---|---|---|
| stringM. . stringN | TRUCKS3. . TRUCKS204 | TRUCK3，TRUCK4，…，TRUCK204 |
| dayM. . dayN | MON. . FRI | MON，TUE，WED，THU，FRI |
| monthM. . monthN | OCT. . JAN | OCT，NOV，DEC，JAN |
| monthYearM. . monthYearN | OCT2001. . JAN2002 | OCT2001，NOV2001，DEC2001，JAN2002 |

当使用 1. . n 格式时，n 可以是任意的正整数，并且起始元素名称必须取 1。stringM. . stringN 的格式允许使用任何字母开头，并且要符合 LINGO 的命名规则，M 和 N 必须为非负整数。dayM. . dayN 允许以一个星期内任意一天的名称命名，所有的名称是其英文单词的前三个字母，可用的选择是 Mon、Tue、Wed、Thu、Fri、Sat 和 Sun。monthM. . monthN 允许以一年内任意一个月的名称命名，所有的名称是其英文单词的前三个字母，可用的选择是 Jan、Feb、Mar、Apr、May、Jun、Jul、Aug、Sep、Oct、Nov 和 Dec。monthYearM. . monthYearN 允许指定一个月和 4 位阿拉伯数字表示的年。

在 Wireless Widgets 公司的运输模型中，集合 WAREHOUSES 可以如下定义：

```
WAREHOUSES / 1..6/ : CAPACITY ;
```

使用 1. . n 间接定义方式时，还可以在数据域中存放其长度，然后在后面的集合域中引用这个长度值。例如：

```
DATA :
  NUMBER _ OF _ WH = 6 ;
ENDDATA
SETS :
  WAREHOUSES / 1..NUMBER _ OF _ WH/ : CAPACITY ;
ENDSETS
```

集合的元素可能有一个或多个属性，这些属性在集合定义中的属性表（attribute _ list）内列出。一个属性值就是集合中每一个元素表现出来的特性。例如，集合 WAREHOUSES 只有一个 CAPACITY 属性，用来代表 WAREHOUSE 的运输能力。属性名称必须遵循标准命名规则，并且用逗号隔开。

为了说明问题，假设仓库有与其位置和装卸码头数量有关的额外属性，这些额外属性可以被加到集合属性表中，如下所示：

```
WAREHOUSES / 1..6/ : CAPACITY, LOCATION, DOCKS ;
```

## 2.3.2 衍生集合的定义

为了定义一个衍生集合，要说明如下参数：

1）集合名称；

2）集合的父集合名称；

3) 集合的元素 (可选) ;

4) 集合的元素属性 (可选) 。

衍生集合在定义时采用如下语法:

```
setname(parent_set_list) [/member_list/] [:attribute_list];
```

setname 是衍生集合的名称。parent_set_list 是先前定义的集合列表,用逗号隔开。如果没有指明一个衍生集合的元素列表 (member_list) ,LINGO 将采用其父集合元素的所有组合作为衍生集合的元素构建新集合。作为一个例子,参看下面的集合域:

```
SETS:
   PRODUCT/A B/;
   MACHINE/M N/;
   WEEK/1..2/;
   ALLOWED(PRODUCT, MACHINE, WEEK);
ENDSETS
```

PRODUCT、MACHINE 和 WEEK 是基本集合,ALLOWED 是从 PRODUCT、MACHINE 和 WEEK 派生出来的集合。ALLOWED 使用了所有父集合中的元素。表 2-2 列出 ALLOWED 集合中的元素。

表 2-2 ALLOWED 集合元素

| 索引 | 元素 | 索引 | 元素 |
|---|---|---|---|
| 1 | (A,M,1) | 5 | (B,M,1) |
| 2 | (A,M,2) | 6 | (B,M,2) |
| 3 | (A,N,1) | 7 | (B,N,1) |
| 4 | (A,N,2) | 8 | (B,N,2) |

元素列表是可选的,当衍生集合只是父集合所有元素组合中的一部分时,才会用到元素列表。此外,元素列表还可以放在模型的数据域中进行说明。

如前所述,如果元素列表被省略,衍生集合将包含父集合中所有元素的组合,这样,它就被认为是一个稠密集合。当衍生集合中包含一个元素列表时,它就被限制为一个稠密集合的子集,称为稀疏集合。

一个衍生集合的列表可以使用以下方法构建:

1) 直接元素列表;

2) 集合元素过滤器。

当使用直接元素列表的方法定义一个衍生集合的元素时,必须具体地列出包含在集合中的所有元素的名称,每一个元素必须来自于由其父集合元素组成的稠密集合中。返回上面的例子,可以利用直接元素列表定义集合:

```
ALLOWED(PRODUCT, MACHINE, WEEK)
/A M 1, A N 2, B N 1/;
```

这样，ALLOWED 集合不是含有所有的 8 个元素，而是只有 (A，M，1)、(A，N，2) 和 (B，N，1) 3 个元素。

如果集合是含有大量元素的稀疏集合，用直接元素列表就显得过于复杂。如果稀疏集合中含有的元素与非集合中的元素有明确的区分条件，就可以使用集合元素过滤器来定义稀疏集合的元素，这样可以节约大量的工作。使用集合元素过滤器定义一个衍生集合时必须指明集合元素应该符合的逻辑条件。这个逻辑条件就是一个过滤器，它能把不符合条件的元素去掉。

假设已经定义了一个卡车集合 TRUCKS，每一辆卡车有一个属性 CAPACITY。要以 TRUCKS 为基础构建一个衍生集合，元素为运输能力大的卡车。可以使用集合元素过滤器的方法：

```
HEAVY_DUTY(TRUCKS) | CAPACITY(&1) #GT# 50000;
```

该衍生集合的名称是 HEAVY_DUTY，竖线 (|) 作为集合元素过滤器的标志。过滤器将运输能力超过 50 000 的卡车放入集合 HEAVY_DUTY 中。符号 &1 是过滤器中的集合索引占位符。当使用集合元素过滤器时，LINGO 要扫描所有父集合中的元素。如果集合元素通过过滤器的验证，就会被放入衍生集合中。在验证时，第一个父集合的元素放到 &1，第二个父集合的元素放到 &2，依此类推。在这个例子中，只有一个父集合 TRUCKS，所以没有用到 &2。符号 #GT# 是一个逻辑运算符，意为“大于”。表 2-3 列出 LINGO 中的几种逻辑运算符。

表 2-3　逻辑运算符

| 逻辑运算符 | 含义 | 逻辑运算符 | 含义 |
|---|---|---|---|
| #EQ# | 等于 | #GT# | 大于 |
| #NE# | 不等于 | #LT# | 小于 |
| #GE# | 大于等于 | #LE# | 小于等于 |

## 2.3.3　总结

LINGO 有两种类型的集合，即基本集合和衍生集合。

基本集合是模型的基本组成部分，不能被分成更小的部分。基本集合可以用直接列表或间接列表定义。使用直接列表时，需要说明每一个元素；使用间接列表时，要输入起始元素和末端元素，LINGO 能够生成中间的其他元素。

衍生集合由其他集合组成，组成它的集合称为衍生集合的父集合。父集合可以是基本集合也可以是衍生集合。衍生集合可以是稠密集合也可以是稀疏集合。稠密集合包含父集合中所有的元素组合，而稀疏集合只包含部分父集合元素的组合。可以用直接列表或过滤器两种方法定义集合元素。使用直接列表时要列出所有稀疏集合中的元素，使用过滤器时则使用逻辑条件确定所有满足条件的元素。各种集合的关系如图 2-1 所示。

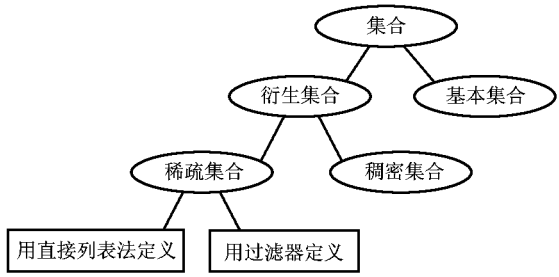



**图 2-1　LINGO 集合类型及关系**

## 2.4　模型的数据域

为了给集合的属性赋值，LINGO 提供了另一个可选择的域——数据域。数据域可以使数据与模型的其他部分分开，这对于模型的维护和修改都有重要的意义。

与集合域类似，数据域以关键字“DATA：”开始，以关键字“ENDDATA”结束。在数据域中，使用以下语句将集合域中定义的属性列表初始化：

```
attribute_list = value_list;
```

属性列表（attribute _ list）中包含需要初始化的属性名，属性名之间可以用空格或逗号隔开。如果在表达式的左边有多于一个的属性名，则这些属性必须是在同一集合中定义的。数值列表（value _ list）中包含将要赋给属性列表中属性的值，同样用空格或逗号隔开。例如：

```
SETS:
   SET1 /A, B, C/:X, Y;
ENDSETS
DATA:
   X = 1 2 3;
   Y = 4 5 6;
ENDDATA
```

上述模型有两个属性 X 和 Y，均在集合 SET1 中被定义。X 的三个值为 1、2、3，Y 的三个值为 4、5、6。还可以用以下的混合数据表达式赋值：

```
SETS:
   SET1 /A, B, C/:X, Y;
ENDSETS
DATA:
   X Y = 1 4
         2 5
         3 6;
ENDDATA
```

当 LINGO 读取数据表中的数据时，先读第一行的第一个数据，再读第二行的第一个数据，

依此类推。换句话说，LINGO 是按列输入，而不是按行输入。

正如前面所提到的，LINGO 还可以在数据域中指定集合元素及其属性值。例如可以把上述模型转换成如下形式：

```
SETS:
   SET1:X, Y;
ENDSETS
DATA:
   SET1 X Y = A 1 4
              B 2 5
              C 3 6;
ENDDATA
```

这样，在模型的数据域中可独立设置所有的数据（集合元素和属性值）。这些数据是按列排列的，恰好与在数据库中的数据排列方式形成镜像关系。

## 2.5 集合循环函数

集合有着强大的功能，用单一的表达式就可以操作集合中的所有元素，在 LINGO 中实现这一功能的函数称为集合循环函数。

集合循环函数可以使所有的元素重复完成一些操作。LINGO 有四种集合循环函数，其名称和功能如表 2-4 所示。

表 2-4 集合循环函数

| 函数 | 函数功能 |
|---|---|
| @FOR | 用于形成作用于集合所有元素的约束条件 |
| @SUM | 用于计算作用于集合中元素的表达式的和 |
| @MIN | 用于计算作用于集合中元素的表达式最小值 |
| @MAX | 用于计算作用于集合中元素的表达式最大值 |

集合循环函数的语法如下：

```
@function(setname [(set_index_list)
        [| conditional_qualifier]] :expression_list);
```

式中，@function 为循环函数的名称；setname 是循环函数操作的集合名；集合索引列表 set_index_list 是可选的，用来构建索引，每一项索引对应集合中的一个元素；条件过滤器 conditional_qualifier 也是可选的，用于控制循环函数的使用范围，当 LINGO 对集合中的元素进行循环时，首先为条件过滤器估值，如果条件过滤器的值为真，函数将对其进行操作，否则就跳过；expression_list 是应用于集合中每个元素的表达式列表。当使用 @FOR 函数时，列表中可以包含许多表达式，彼此间用分号隔开。这样，这些表达式就以约束条件的形式加入到模型

中。当使用其他三个循环函数(@SUM、@MAX 和 @MIN)时,表达式列表中只能包含一个表达式。如果索引列表被忽略,所有在表达式列表中被提到的属性都必须是在同一集合 setname 中被定义的。

## 2.5.1 @SUM 函数

先参考如下模型:

```
SETS:
   VENDORS / V1 V2 V3 V4 V5 /:DEMAND;
ENDSETS
DATA:
   DEMAND = 5 1 3 4 6;
ENDDATA
```

集合 VENDORS 中的元素具有属性 DEMAND,可以在 ENDDATA 语句后加一个表达式求属性 DEMAND 的和,如下所示:

```
TOTAL_DEMAND = @SUM(VENDORS(J):DEMAND(J));
```

LINGO 通过一个内部的累加器对 @SUM 函数进行初始化,赋值为 0;然后,LINGO 开始对集合 VENDORS 的所有元素进行循环,集合索引变量 J 指向 VENDORS 的第一个元素 V1,并将 DEMAND(V1)添加到累加器里,这一过程持续到所有 DEMAND 的值都被加到累加器中;最后,DEMAND 的和被储存到变量 TOTAL_DEMAND 中。

在此例中,表达式列表中的所有属性都是在集合 VENDORS 中定义的,所以求和表达式可以写成以下形式:

```
TOTAL_DEMAND = @SUM(VENDORS:DEMAND);
```

在上述情况下,舍去了多余的集合元素索引以及 DEMAND 的索引,这时称索引被隐藏了。如果表达式列表中的属性属于不同的父集合,则不允许使用隐藏索引。

另外,如果想求属性 DEMAND 中前三个元素的和,需要在集合索引后加入一个条件过滤器:

```
DEMAND_3 = @SUM(VENDORS(J)|J #LE# 3:DEMAND(J));
```

在 LINGO 计算求和时,首先将索引变量 J 送入条件过滤器 J #LE# 3,如果返回值为真,DEMAND(j)的值就被加入到和中。最后的结果是 LINGO 将 DEMAND 的前三项求和,而忽略掉第四项和第五项,总和为 9。

## 2.5.2 @MIN 和 @MAX 函数

@MIN 和 @MAX 函数用于寻找集合中元素表达式的最大和最小值。

在前面的模型中,为了找到 DEMAND 的最大和最小值,需要加入以下两个表达式:

```
MIN_DEMAND = @MIN(VENDORS(J):DEMAND(J));
MAX_DEMAND = @MAX(VENDORS(J):DEMAND(J));
```

在本例的@MIN和@MAX函数中,同样可以使用隐藏索引:

```
MIN_DEMAND = @MIN(VENDORS:DEMAND);
MAX_DEMAND = @MAX(VENDORS:DEMAND);
```

求解这个模型后,LINGO返回DEMAND的最小值(MIN_DEMAND)和最大值(MAX_DEMAND)分别为1.000 000和6.000 000。

同样,可以使用条件过滤器计算出DEMAND的前三个元素的最小值和最大值:

```
MIN_DEMAND3 =
   @MIN(VENDORS(J) | J #LE# 3:DEMAND(J));
MAX_DEMAND3 =
   @MAX(VENDORS(J) | J #LE# 3:DEMAND(J));
```

结果分别为1.000 000和5.000 000。

## 2.5.3 @FOR函数

@FOR函数用于对集合元素产生约束条件。如前所述,如果使用标量变量,就要逐个输入所有的约束,而@FOR函数允许一组相同的约束只输入一次,LINGO会针对集合中的每一个元素产生一个约束。因此,@FOR语句为用户提供了一个强大的工具。

为了说明@FOR函数的用法,参看下面的集合定义:

```
SETS:
   TRUCKS/MAC, PETERBILT, FORD, DODGE/:HAUL;
ENDSETS
```

这个4辆卡车的集合只有一个属性HAUL。如果HAUL表示卡车的载货量,可以用@FOR函数限制每辆卡车的载货量不能超过2 500磅,表达式为:

```
@FOR(TRUCKS(T):HAUL(T) <= 2500);
```

在这种情况下,检查LINGO采用此表达式生成的约束条件对于调试程序很有帮助。在Windows系统下,可以通过运行LINGO|Generate|Display|model命令实现这一操作,在其他操作系统下则使用GENERATE命令实现。运行这个命令后,发现LINGO为该模型添加了4个约束条件:

```
HAUL(MAC) <= 2500
HAUL(PETERBILT) <= 2500
HAUL(FORD) <= 2500
HAUL(DODGE) <= 2500
```

换句话说,LINGO为每一辆卡车设置了一个约束,限制它的载货量不超过2 500磅。

下面的这个模型用@FOR函数来计算属性VALUE中5个数的倒数:

```
SETS:
   NUMBERS/1..5/:VALUE, RECIPROCAL;
ENDSETS
```

```
DATA:
   VALUE = 3 4 2 7 10;
ENDDATA
   @FOR(NUMBERS(I):
      RECIPROCAL(I) = 1/VALUE(I));
```

求解上面的模型,得到 5 个倒数值:

```
RECIPROCAL(1)        0.3333333
RECIPROCAL(2)        0.2500000
RECIPROCAL(3)        0.5000000
RECIPROCAL(4)        0.1428571
RECIPROCAL(5)        0.1000000
```

由于 0 的倒数没有意义,所以还可以在 @FOR 函数中使用一个条件过滤器跳过求 0 的倒数:

```
@FOR(NUMBERS(I) | VALUE(I) #NE# 0:
   RECIPROCAL(I) = 1/VALUE(I));
```

### 2.5.4 嵌套的集合循环函数

在大型模型中,可能遇到使用其他循环函数的循环。当在一个集合循环函数中使用另一个循环函数时,就称为集合循环函数的嵌套。

在 Wireless Widgets 公司的运输模型中可以发现嵌套循环的使用。每一个销售商需要的工艺品数量都要被满足,用下列 LINGO 语句实现这个需求约束:

```
@FOR(VENDORS(J):
 @SUM(WAREHOUSES(I):VOLUME(I,J)) =
 DEMAND(J));
```

这组约束条件表示:对于每一个销售商,累加所有仓库到该销售商的运输量的总和,并使其与需求量相等。在这种情况下,需要在 @FOR 函数中嵌套 @SUM 函数。@SUM、@MAX 和 @MIN 函数可以嵌套在任意循环函数中,但是,@FOR 函数只能嵌套在 @FOR 函数中。

### 2.5.5 总结

本节介绍了循环函数的强大功能,利用它可以减少很多工作量。如果不使用集合和循环函数,建立模型会很困难,而且随着模型的增大,工作量会成倍增加。

现在已经知道如何创建集合,如何利用数据域将集合初始化,如何利用循环函数处理集合。下面将举例说明集合在模型中的应用情况。

## 2.6 基于集合的模型举例

### 2.6.1 基本集合模型举例——员工安排模型

员工安排模型主要说明基本集合的使用方法,目的是计算出满足一周内每天的员工需求

时,最少需要的员工总数。

这个例子中的模型可以在 LINGO 主目录下的子目录 SAMPLES 中找到,名为 STAFFDEM。

#### 2.6.1.1 员工安排问题

假设经营一家餐馆,一周营业 7 天。餐馆雇用的员工每周工作 5 天,休息 2 天,并且每个员工得到相同的周薪。基于过去的经验,一周内有些天比较忙而有些天顾客相对较少,所以每天需要的员工数也不同,如表 2-5 所示。

表 2-5 员工需求量

| 星期 | 一 | 二 | 三 | 四 | 五 | 六 | 日 |
|---|---|---|---|---|---|---|---|
| 需求量 (人) | 20 | 16 | 13 | 16 | 19 | 14 | 12 |

该问题就是要在满足每天员工需求量的条件下确定所需雇用的最少员工总数。

#### 2.6.1.2 建立模型

当使用集合建立模型时,首先考虑的问题是相关的集合和属性是什么。在这个模型中,涉及一个基本集合,即一周中的每一天,把它命名为 DAYS,相应地写出集合域:

```
SETS:
   DAYS / MON, TUE, WED, THU, FRI, SAT, SUN/;
ENDSETS
```

或者间接定义该集合:

```
SETS:
   DAYS / MON..SUN/;
ENDSETS
```

集合 DAYS 有两个属性,一个是每天的员工需求数,另一个是每天开始工作的员工数。把这两个属性分别命名为 REQUIRED 和 START,并将其加入到数据域中:

```
SETS:
   DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
```

集合和属性表定义完毕后,就要确定哪些属性是已知数据,哪些属性是决策变量。在这个模型中,REQUIRED 是已知数据,START 是决策变量。然后,在数据域中为 REQUIRED 属性赋值:

```
DATA:
   REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
```

接下来,列出模型的目标函数和约束条件。该模型的目标是使雇用的员工数最少:

```
MIN = @SUM(DAYS(I):START(I));
```

该模型中仅有一个约束条件,即必须使每天工作的员工数满足或是超过每天的需求量。而当天工作的员工数可以按照下式计算:

当天工作的员工数 = (当天开始工作的人数) + (一天前开始工作的员工数) + (两天前开始工作的员工数) + (三天前开始工作的员工数) + (四天前开始工作的员工数)

换句话说,为了算出当天工作的人数,就需要求出当天开始工作的人数和前四天开始工作的人数之和,而五天和六天前开始工作的人数不计入,因为这些员工正在休息。用数学符号表达的约束条件为:

$$\sum i = j-4, j\ \mathrm{START}i \geqslant \mathrm{REQUIRED}\ j\ ,\ \text{for}\ j \in \mathrm{DAYS}$$

转换成 LINGO 语言为:

```
@FOR(DAYS(J):
@SUM(DAYS(I)|I#LE#5:START(J-I+1))
>=REQUIRED(J)
);
```

求解该模型时,会遇到如图 2-2 所示错误信息。

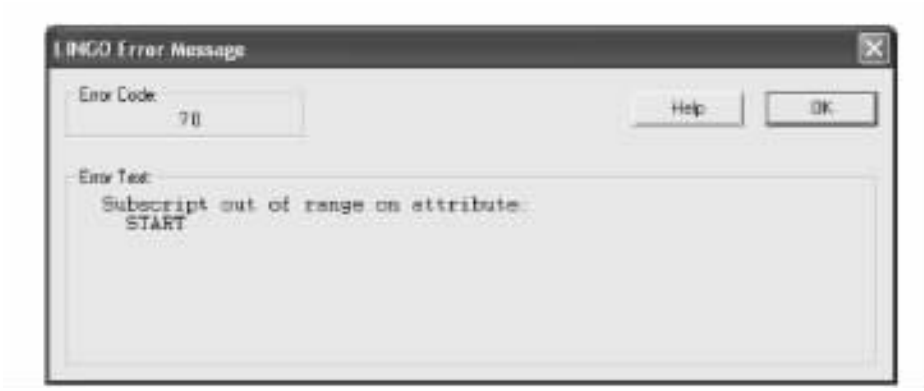



图 2-2 错误信息

以星期四为例解释为什么会出现这个错误。星期四在集合 DAYS 中的索引是 4,并且星期四的约束条件为:

```
START(4-1+1)+START(4-2+1)+
START(4-3+1)+START(4-4+1)+
START(4-5+1)>=REQUIRED(4);
```

化简后得到:

```
START(4)+START(3)+
START(2)+START(1)+
START(0)>=REQUIRED(4);
```

START(0)是错误的根源,因为0索引"超出了范围(out of range)",希望小于或等于0的索引回绕到一周的末端。也就是说,0对应7,即Sunday;-1对应6,即Saturday,等等。LINGO中有一个函数@WRAP可以解决上述问题。

函数@WRAP有两个参数INDEX和LIMIT,其返回值为J,并且

$$J = INDEX - K * LIMIT$$

式中,K为能使J落入范围[1,LIMIT]的整数。也就是说,函数@WRAP将在INDEX上减去或加上数个LIMIT,直到INDEX落入[1,LIMIT]区间内。这样就解决了建立多阶段计划模型时需要回绕索引的问题。

加入函数@WRAP后得到最终正确的约束条件:

```
@FOR(DAYS(J):
   @SUM(DAYS(I)|I#LE#5:
      START(@WRAP(J-I+1,7)))
         >=REQUIRED(J)
);
```

### 2.6.1.3 结果

下面是完整的员工安排模型:

```
SETS:
   DAYS/MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
DATA:
   REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
MIN = @SUM(DAYS(I):START(I));
@FOR(DAYS(J):
   @SUM(DAYS(I)|I#LE#5:
      START(@WRAP(J-I+1,7)))
         >=REQUIRED(J)
);
```

在LINGO中求解这个模型,得到如下结果:

```
Optimal solution found at step:        8
Objective value:                    22.00000
  Variable        Value        Reduced Cost
START(MON)      8.000000         0.0000000
START(TUE)      2.000000         0.0000000
START(WED)     0.0000000         0.0000000
START(THU)      6.000000         0.0000000
```

```
START (FRI)       3.000000          0.0000000
START (SAT)       3.000000          0.0000000
START (SUN)       0.0000000         0.0000000
  Row    Slack or Surplus    Dual Price
    1         22.00000          1.000000
    2        0.0000000         -0.2000000
    3        0.0000000         -0.2000000
    4        0.0000000         -0.2000000
    5        0.0000000         -0.2000000
    6        0.0000000         -0.2000000
    7        0.0000000         -0.2000000
    8        0.0000000         -0.2000000
```

可见,目标函数值为 22,表示共需雇用 22 个员工,并且按照表 2-6 安排员工的工作。

表 2-6　员工安排结果

| 星期 | 一 | 二 | 三 | 四 | 五 | 六 | 日 |
|---|---|---|---|---|---|---|---|
| 开始工作员工数 (人) | 8 | 2 | 0 | 6 | 3 | 3 | 0 |

从结果中的员工需求约束行还能看到,每天的松弛变量都是 0,这意味着员工数刚好满足需求。

## 2.6.2　稠密衍生集合模型举例——配比模型

在这一模型中,将说明如何使用稠密衍生集合解决配比问题。在配比问题中,最终产品由几种原材料组成,对于每一种原材料的一个或几个性质都有最低的质量标准。该问题的目标就是在满足质量要求的情况下使费用最低。

这个模型可以在 LINGO 主目录下的子目录 SAMPLES 中找到,名为 CHESS。

### 2.6.2.1　配比模型

Chess 公司生产四种品牌的混合坚果。这四种品牌的混合坚果分别为 Pawn、Knight、Bishop 和 King。每一种品牌的产品包含不同比例的花生和腰果,价格也有所不同,如表 2-7 所示。

表 2-7　产品价格

| | Pawn | Knight | Bishop | King |
|---|---|---|---|---|
| 花生 (盎司) | 15 | 10 | 6 | 2 |
| 腰果 (盎司) | 1 | 6 | 10 | 14 |
| 售价 (磅) | 2 | 3 | 4 | 5 |

已知 Chess 公司每天可以从供货商处获得 750 磅花生和 250 磅腰果。该问题的目标是在

不超过可得到的坚果数的情况下,确定每天生产多少磅各种产品可以获得最大的利润。

#### 2.6.2.2 建立模型

在配比模型中有两个基本集合,分别为坚果类型 NUTS 和每种品牌的产品 BRANDS,把它们列入模型的集合域中:

```
SETS:
   NUTS / PEANUTS, CASHEWS/:SUPPLY;
   BRANDS / PAWN, KNIGHT, BISHOP, KING/:
    PRICE, PRODUCE;
ENDSETS
```

集合 NUTS 有唯一的属性 SUPPLY,表示每天的坚果供应量。集合 BRANDS 有 PRICE 和 PRODUCE 两个属性,分别表示每种品牌产品的售价和每天生产每种品牌产品的数量。

此外,还需要建立一个由集合 NUTS 和 BRANDS 形成的衍生集合,用于表示每种品牌产品的坚果组成。加入此集合,完成模型的集合域如下:

```
SETS:
   NUTS / PEANUTS, CASHEWS/:SUPPLY;
   BRANDS / PAWN, KNIGHT, BISHOP, KING/:
    PRICE, PRODUCE;
 FORMULA(NUTS, BRANDS):OUNCES;
ENDSETS
```

这个衍生集合名为 FORMULA,它有一个属性 OUNCES,表示一磅每种品牌的产品需要多少盎司的花生和腰果。因为没有对这个衍生集合进行特别定义,LINGO 将坚果和品牌的所有配对形式组成的元素当作集合的元素,所以该集合为稠密衍生集合。

集合定义完成后,在数据域中对三个属性 SUPPLY、PRICE 和 OUNCES 进行赋值:

```
DATA:
   SUPPLY = 750  250;
   PRICE = 2  3  4  5;
   OUNCES = 15  10  6  2
            1  6  10  14;
ENDDATA
```

接下来,建立该问题的目标函数和约束条件。目标函数直接表达为收益的最大化:

```
MAX = @SUM(BRANDS(I):
 PRICE(I) * PRODUCE(I));
```

该模型只有一类约束条件,就是每天使用的坚果数量不能超过每天的供应量:

```
@FOR(NUTS(I):
 @SUM(BRANDS(J):
  OUNCES(I,J) * PRODUCE(J) / 16) < =
```

```
    SUPPLY(I)
);
```

**注 左边的和除以 16 是将盎司转化为磅。**

### 2.6.2.3 结果

**下面是完整的配比模型：**

```
SETS:
   NUTS / PEANUTS, CASHEWS/ : SUPPLY;
     BRANDS / PAWN, KNIGHT, BISHOP, KING/ :
    PRICE, PRODUCE;
   FORMULA(NUTS, BRANDS) : OUNCES;
ENDSETS
DATA:
   SUPPLY = 750 250;
   PRICE = 2 3 4 5;
   OUNCES = 15 10  6  2
            1  6 10 14;
ENDDATA
MAX = @SUM(BRANDS(I) :
 PRICE(I) * PRODUCE(I));
@FOR(NUTS(I) :
 @SUM(BRANDS(J) :
  OUNCES(I, J) * PRODUCE(J) / 16) < =
   SUPPLY(I)
);
```

**在 LINGO 中求解这个模型,得到如下结果：**

```
Optimal solution found at step:         2
 Objective value:                 2692.308
      Variable        Value        Reduced Cost
   PRODUCE(PAWN)     769.2308        0.0000000
 PRODUCE(KNIGHT)     0.000000        0.1538461
 PRODUCE(BISHOP)     0.000000        0.7692297
   PRODUCE(KING)     230.7692        0.0000000
     Row      Slack or Surplus      Dual Price
      1             2692.308         1.000000
      2             0.000000         1.769231
      3             0.000000         5.461538
```

**模型结果表示每天生产 769.2 磅的 Pawn 产品和 230.8 磅的 King 产品能使总收入最高,为 $ 2 692.30。另外,结果报告中还提供了一些有用的信息 :影子价格表示 Chess 公司愿意以每**

磅 $1.77 的价格额外购买花生，以每磅 $5.46 的价格额外购买腰果。如果由于市场原因，Chess 公司至少要生产 Knight 和 Bishop 中的一种，Reduced Cost 值指出，每生产一磅 Knight 产品总收益会减少 15.4 美分，而每生产一磅 Bishop 产品总收益会减少 76.9 美分。

## 2.6.3 稀疏衍生集合模型举例——直接列表

在这个例子中，使用直接列表定义稀疏衍生集合，即把集合中的每一个元素明确地列出来。

在此例中，建立一个 PERT（Project Evaluation and Review Technique，工程评价和审核技术）模型来确定一个新产品的生产流程中的关键作业路线。PERT 模型产生于 20 世纪 50 年代，是一个简单而功能强大的模型，能够帮助管理者追踪大型工程。PERT 模型对于识别一个工程中的关键工序非常有用。如果关键工序被延迟，整个工程就要被延迟。

在 LINGO 主目录下的子目录 SAMPLES 中包含这个模型的公式，名为 PERT。

### 2.6.3.1 工程安排问题

Wireless Widgets 公司将要下线一种新产品 Solar Widget。为了使这种产品按时下线，WW 公司想用 PERT 分析什么是完成这一任务的关键。在产品推广之前必须完成的任务以及它们的预期完成工期列于表 2-8 中。

表 2-8 工序及完成所需工期

| 工序 | 工期（周） | 工序 | 工期（周） |
|---|---|---|---|
| 确定方案 | 10 | 生产安排 | 7 |
| 需求预测 | 14 | 成本核算 | 4 |
| 竞争调查 | 3 | 训练销售人员 | 10 |
| 设定价格 | 3 | | |

有些工序必须在其他工序开始前完成，图 2-3 表示出这种关系。例如，起始于需求预测节点的两个箭头表示，在需求预测完成之后生产安排和设定价格两个工序才能开始。

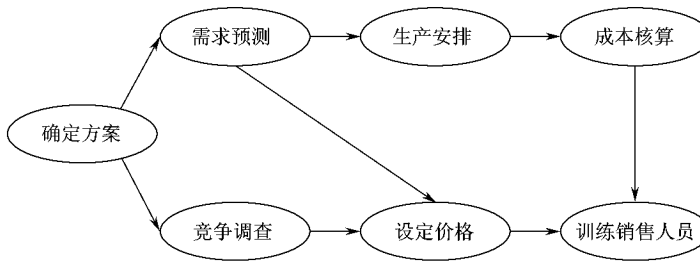



图 2-3 生产工序优先关系图

该问题的目的是为 Solar Widget 的推广建立一个 PERT 模型，分析任务的关键工序。

2.6.3.2 建立模型

在建立模型时,首先需要使用一个基本集合 TASKS 来表示这个工程中的各项工序。把以下的集合定义加入到模型中:

```
TASKS / DESIGN, FORECAST, SURVEY, PRICE,
    SCHEDULE, COSTOUT, TRAIN/ : TIME, ES, LS, SLACK;
```

集合 TASKS 有四个属性,其定义分别为:

TIME——完成工序所需的时间;

ES——工序的最早开始时间;

LS——工序的最晚开始时间;

SLACK——工序的 LS 和 ES 之间的差别,即松弛时间。

属性 TIME 以已知数据的形式给出,剩下的三个属性值则可以通过计算获得。如果某个工序的松弛时间为 0,意味着这个工序必须按时完成,否则整个工程就要延期,所有这样的工序构成了工程的关键工序。

为了计算出所有工序的开始时间,必须使用优先关系表,并将优先关系输入到模型中。优先关系可以被看作是排列好的工序列表。例如,工序 DESIGN 必须在工序 FORECAST 之前完成,也可以表示为有序对 (DESIGN,FORECAST)。利用集合 TASKS 构建一个二维表来输入优先关系列表,并加入集合定义中:

```
PRED (TASKS, TASKS) /
 DESIGN,FORECAST,
 DESIGN,SURVEY,
 FORECAST,PRICE,
 FORECAST,SCHEDULE,
 SURVEY,PRICE,
 SCHEDULE,COSTOUT,
 PRICE,TRAIN,
 COSTOUT,TRAIN /;
```

需要注意的是,集合的第一个元素是有序对 (DESIGN,FORECAST),而不是单一的任务 DESIGN。因此集合中有 8 个元素,所有这些元素都源于优先关系图。

可见,集合 PRED 是使用直接列表方法定义的稀疏衍生集合,它是由两个 TASKS 集合的元素交叉配对衍生出来的集合的子集。它是稀疏集合的原因为它只包含所有 49 个元素中的 8 个。

然后,在模型的数据域中对已知数据 TIME 进行赋值:

```
DATA:
   TIME = 10, 14, 3, 3, 7, 4, 10;
ENDDATA
```

这样,就可以计算其他属性的值。首先列出计算 ES 的公式。一项任务只能在其紧前工序

完成后才能进行,所以,如果知道了一项工序所有紧前工序的最迟完工时间,也就知道了此项工序的最早开工时间。工序 t 的最早开工时间等于所有紧前工序的完成时间与各自最早开工时间的和的最大值。用 LINGO 语言表示为:

```
@FOR(TASKS(J)|J #GT# 1:
 ES(J) = @MAX(PRED(I,J):ES(I) + TIME(I))
);
```

因为第一道工序没有紧前工序,所以通过添加条件过滤器 J #GT# 1 跳过了计算的第一道工序,并可以给第一道工序赋一个任意的开始时间。

计算 LS 比较复杂,但是与 ES 类似。工序 t 的最晚开工时间是所有的后续工序的最早开工时间与工序 t 的完成时间之差的最小值。如果工序 t 晚于此时间开工,就会影响到后面工序的最早开工时间,表达为下列 LINGO 语句:

```
@FOR(TASKS(I)|I #LT# LTASK:
 LS(I) = @MIN(PRED(I,J):LS(J) - TIME(I))
);
```

因为最后一道工序没有后续的工序,所以在此省略了最后一道工序的计算。接下来计算松弛时间,即求 LS 和 ES 之差,可以表示为:

```
@FOR(TASKS(I):SLACK(I) = LS(I) - ES(I));
ES(1) = 0;
```

至此,已经计算出除最后工序的最晚开工时间之外的所有变量值。如果最后一道工序晚于最早开工时间开始,整个工程就会被延误。所以,定义最后一道工序的最晚开工时间等于它的最早开工时间。在 LINGO 中表示为:

```
LS(7) = ES(7);
```

这样虽然可以实现,但不是表达这种关系的最好方式。假如向模型中添加一些工序,就必须改正上述等式中的"7"。基于模型语言建立的 LINGO 模型应该独立于数据,最好采用如下表达方式:

```
LTASK = @SIZE(TASKS);
LS(LTASK) = ES(LTASK);
```

@SIZE 函数的返回值为集合的大小。在这个例子中,返回值为 7。这样,如果改变工序的数量,@SIZE 将会返回新的值。因此也就保持了数据与模型相互独立。

#### 2.6.3.3 结果

整个 PERT 模型和部分结果如下:

```
SETS:
   TASKS / DESIGN, FORECAST, SURVEY, PRICE,
    SCHEDULE, COSTOUT, TRAIN/:TIME, ES, LS, SLACK;
   PRED(TASKS, TASKS) /
```

```
      DESIGN,FORECAST,
      DESIGN,SURVEY,
      FORECAST,PRICE,
      FORECAST,SCHEDULE,
      SURVEY,PRICE,
      SCHEDULE,COSTOUT,
      PRICE,TRAIN,
      COSTOUT,TRAIN / ;
   ENDSETS
   DATA :
      TIME = 10, 14, 3, 3, 7, 4, 10 ;
   ENDDATA
   @ FOR (TASKS (J) | J # GT # 1 :
   ES (J) = @ MAX (PRED (I, J) : ES (I) + TIME (I))
   ) ;
   @ FOR (TASKS (I) | I # LT # LTASK :
   LS (I) = @ MIN ( PRED (I, J) : LS (J) - TIME (I)) ;
   ) ;
   @ FOR (TASKS (I) : SLACK (I) = LS (I) - ES (I)) ;
   ES (1) = 0 ;
   LTASK = @ SIZE (TASKS) ;
LS (LTASK) = ES (LTASK) ;

   Feasible solution found at step: 0
      Variable                Value
         LTASK              7.000000
     ES (DESIGN)            0.0000000
   ES (FORECAST)             10.00000
     ES (SURVEY)             10.00000
      ES (PRICE)             24.00000
   ES (SCHEDULE)             24.00000
    ES (COSTOUT)             31.00000
      ES (TRAIN)             35.00000
     LS (DESIGN)            0.0000000
   LS (FORECAST)             10.00000
     LS (SURVEY)             29.00000
      LS (PRICE)             32.00000
   LS (SCHEDULE)             24.00000
    LS (COSTOUT)             31.00000
      LS (TRAIN)             35.00000
  SLACK (DESIGN)            0.0000000
SLACK (FORECAST)            0.0000000
```

| | |
|---|---|
| SLACK(SURVEY) | 19.00000 |
| SLACK(PRICE) | 8.000000 |
| SLACK(SCHEDULE) | 0.0000000 |
| SLACK(COSTOUT) | 0.0000000 |
| SLACK(TRAIN) | 0.0000000 |

要注意工序的松弛时间,例如工序 SURVEY 和 PRICE 的松弛时间分别为 19 个星期和 8 个星期。它们的最早开工时间只要在松弛时间内取值,就不会影响整个工程的进度。

另一方面,由于工序 DESIGN、FORECAST、SCHEDULE、COSTOUT 和 TRAIN 的松弛变量均为 0,所以这些工序就构成了工程的关键工序。如果这些工序的开工时间延迟,整个工程进度就会延误。管理者要特别关注关键工序的最早开工时间和完成工序所需时间。

最后,ES(TRAIN)的值为 35,表明新产品从生产到下线需要 45 周的时间,即从 35 周开始训练员工,完成训练要 10 周。

## 2.6.4 稀疏衍生集合模型举例——元素过滤器

在这个例子中,用元素过滤器来定义稀疏衍生集合。用元素过滤器是定义稀疏衍生集合的第三种方法。使用此方法定义一个集合时,必须指定集合中的元素需要满足的逻辑条件。

在此例中将建立一个匹配模型。在匹配模型中,假设有 N 个对象要以最小的成本相互匹配,并且数对(I,J)与数对(J,I)相同,规定 I 小于 J。所以,I 和 J 形成的数对组成的集合中只需要 I 小于 J 形成的数对。这一条件可以由集合元素过滤器完成。

在 LINGO 主目录下的子目录 SAMPLES 中包含这个模型文件,名为 MATCHD。

### 2.6.4.1 匹配模型

假设一个公司的策划部有 8 个分析专家,打算搬到一个新的办公地点。新的工作地点有 4 个办公室,需要将专家分成 4 组,每组两个人使用一间新的办公室。

通过以前的观察,发现一些专家喜欢在一起合作,而有些则不然。为了部门的利益,需要找出合适的分配方案,使发生冲突的可能性最小。为了达到这个目标,要列出每一对专家不相容的等级。不相容等级从 1 到 10。1 表示两个专家相处得很好,10 代表两个专家不可能相安无事。通过观察,得到的专家不相容的等级如表 2-9 所示。

表 2-9 专家不相容等级

| 专家 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | — | 9 | 3 | 4 | 2 | 1 | 5 | 6 |
| 2 | — | — | 1 | 7 | 3 | 5 | 2 | 1 |
| 3 | — | — | — | 4 | 4 | 2 | 9 | 2 |
| 4 | — | — | — | — | 1 | 5 | 5 | 2 |
| 5 | — | — | — | — | — | 8 | 7 | 6 |
| 6 | — | — | — | — | — | — | 2 | 3 |
| 7 | — | — | — | — | — | — | — | 4 |

因为专家 I 与专家 J 的配对和专家 J 与专家 I 的配对相同,所以只保留了表格对角线上方的内容。该模型的目的是找出一组专家匹配的方案,使专家匹配的不相容等级的总和最小。

#### 2.6.4.2 建立模型

首先建立一个由 8 个专家组成的基本集合:

```
ANALYSTS/1..8/;
```

最终希望建立的是包括所有可能配对形式的一个衍生集合,由 ANALYSTS 自身的相互交叉产生。所以,首先建立一个稠密集合:

```
PAIRS(ANALYSTS, ANALYSTS);
```

在 PAIRS(I, J)形成的集合中只需要 J > I 的元素。方法是在最终的衍生集合中用一个集合元素过滤器,使数对(I,J)满足 J 大于 I 的条件:

```
PAIRS(ANALYSTS, ANALYSTS) | &2 #GT# &1;
```

元素过滤器由一个竖线标记(|)开始,符号 &1 和 &2 表示集合占位符,占位符只在过滤器中有效。本例中元素过滤器的工作原理是:当 LINGO 建立集合 PAIRS 后,将产生 ANALYSTS 集合自身相互交叉产生的组合 PAIRS(I,J),I、J 分别由符号 &1、&2 取代。每一对组合都将在过滤器中测试,LINGO 为过滤器估值。如果过滤器中的值为真,则数对(I,J)加入集合中。从表格形式来看,留下的是等级表中上三角内的数据。

集合 PAIRS 有两个属性,第一个表示不相容的等级,第二个表示专家 I 与专家 J 是否匹配。这两个属性名为 RATING 和 MATCH,把它们加入集合 PAIRS 的定义中:

```
PAIRS(ANALYSTS, ANALYSTS) | &2 #GT# &1:
    RATING, MATCH;
```

用数据域初始化代表不相容等级的 RATING 属性:

```
DATA:
  RATING =
     9  3  4  2  1  5  6
        1  7  3  5  2  1
           4  4  2  9  2
              1  5  5  2
                 8  7  6
                    2  3
                       4;
ENDDATA
```

属性 MATCH 包含模型中的决策变量,令 MATCH(I,J)取值为 1,表示专家 I 与专家 J 匹配,否则为 0。模型的目标是最终匹配形式的不相容等级之和最小:

```
MIN = @SUM(PAIRS(I, J):
```

```
RATING(I, J) * MATCH(I, J));
```

模型只有一类约束条件,即对于一个专家,只能与另外一个确定的专家匹配。用 LINGO 语句表达这个约束为:

```
@FOR(ANALYSTS(I):
 @SUM(PAIRS(J, K) | J #EQ# I #OR# K #EQ# I:
  MATCH(J, K)) = 1
);
```

约束中用到了条件过滤器 J #EQ# I #OR# K #EQ# I 以及 @SUM 函数。对于每一个专家 I,求出所有含有 I 的 MATCH 变量的和,并使其等于 1,这样就保证了专家 I 只与一个专家匹配。条件过滤器保证仅求出含有变量 I 的 MATCH 的和。

此模型的另一个特点是,当 I 与 J 匹配时就使变量 MATCH(I,J) 的值为 1,否则为 0。特殊情况下,LINGO 变量可以从 0 取到无穷大。因为 MATCH 被限制在 0 和 1,所以需要将变量限定函数 @BIN 添加到属性 MATCH 中。

变量限定函数 @BIN 不像约束那样向模型中加入不等式,而是将变量限制取 0 或 1。当模型中含有二进制变量时,模型被称作整数规划模型。求解整数规划模型比求其他连续性模型困难,对于一些大型模型,可能永远也解不出来。因此,要限制使用二进制变量。为了把 @BIN 函数加入属性表 MATCH 中,用 @FOR 函数表达式:

```
@FOR(PAIRS(I, J):@BIN(MATCH(I, J)));
```

#### 2.6.4.3 结果

匹配模型的整个模型和部分计算结果如下:

```
SETS:
   ANALYSTS/1..8/;
   PAIRS(ANALYSTS, ANALYSTS) | &2 #GT# &1:
    RATING, MATCH;
ENDSETS
DATA:
   RATING =
      9  3  4  2  1  5  6
         1  7  3  5  2  1
            4  4  2  9  2
               1  5  5  2
                  8  7  6
                  2  3
                     4;
ENDDATA
MIN = @SUM(PAIRS(I, J):
 RATING(I, J) * MATCH(I, J));
```

```
@FOR(ANALYSTS(I):
 @SUM(PAIRS(J,K)|J #EQ# I #OR# K #EQ# I:
  MATCH(J,K)) = 1
);
@FOR(PAIRS(I,J):@BIN(MATCH(I,J)));
Optimal solution found at step:        9
 Objective value:                6.000000
 Branch count:                          0
  Variable        Value        Reduced Cost
 MATCH(1,2)    0.0000000        9.000000
 MATCH(1,3)    0.0000000        3.000000
 MATCH(1,4)    0.0000000        4.000000
 MATCH(1,5)    0.0000000        2.000000
 MATCH(1,6)     1.000000        1.000000
 MATCH(1,7)    0.0000000        5.000000
 MATCH(1,8)    0.0000000        6.000000
 MATCH(2,3)    0.0000000        1.000000
 MATCH(2,4)    0.0000000        7.000000
 MATCH(2,5)    0.0000000        3.000000
 MATCH(2,6)    0.0000000        5.000000
 MATCH(2,7)     1.000000        2.000000
 MATCH(2,8)    0.0000000        1.000000
 MATCH(3,4)    0.0000000        4.000000
 MATCH(3,5)    0.0000000        4.000000
 MATCH(3,6)    0.0000000        2.000000
 MATCH(3,7)    0.0000000        9.000000
 MATCH(3,8)     1.000000        2.000000
 MATCH(4,5)     1.000000        1.000000
 MATCH(4,6)    0.0000000        5.000000
 MATCH(4,7)    0.0000000        5.000000
 MATCH(4,8)    0.0000000        2.000000
 MATCH(5,6)    0.0000000        8.000000
 MATCH(5,7)    0.0000000        7.000000
 MATCH(5,8)    0.0000000        6.000000
 MATCH(6,7)    0.0000000        2.000000
 MATCH(6,8)    0.0000000        3.000000
 MATCH(7,8)    0.0000000        4.000000
```

**从目标值中可以看到,所有数对的不相容等级之和为 6。并且由第一列可知,优先匹配的专家为 (1,6)、(2,7)、(3,8) 和 (4,5)。**

## 2.7 总结

在本章中，讨论了集合的概念、定义集合的方法以及集合在模型中的重要作用。通过本章的学习，可以了解在模型的集合域中定义基本集合和衍生集合、构建衍生集合的方法以及各种集合之间的关系，还可以了解在模型的数据域中为集合赋值的方法。此外，本章还着重介绍了循环函数的用法，并通过一些基于集合的模型举例，使读者掌握利用集合和循环函数构建自己模型的方法。

# 第 3 章　变量限定函数的使用

## 3.1　简介

在默认情况下，LINGO 模型中的变量是连续、非负的，即变量的值可以取从零到正无穷大的任意实数。但是，在很多情况下，这是不符合实际需要的。例如，一个变量可能需要取负值或纯整数值。为了不受变量默认值的影响，LINGO 提供了四种变量限定函数。这些函数的名称和主要功能如下：

1）@GIN，限定变量只能取整数；

2）@BIN，限定变量只能取二进制的值（取值为 0，1）；

3）@FREE，允许一个变量取正或负的任意实数值；

4）@BND，限制一个变量只能在一定范围内取值。

本章将通过一些实例介绍如何使用这些函数。

## 3.2　整型变量

LINGO 允许用户定义两种类型的整型变量——普通型和二进制型。一个普通整型变量可以取所有的正整数，一个二进制整型变量可以取 0 和 1。当模型中有一个或多个整型变量时，该模型被认为是一个整数规划（IP）问题。

在许多模型的设计过程中，可能会遇到 Yes/No 的选择问题。例如，有些模型中包含生产/不生产、经营工厂/关闭工厂、从工厂 J 向消费者 I 供应产品/不从工厂 J 向消费者 I 供应产品、进行一项固定资产投资/不进行此项固定资产投资，等等。处理这种 Yes/No 类型的决策变量的标准方法是使用二进制整型变量。

而当舍入结果会出现问题时，就要使用普通整型变量。例如，在一个关于蜡笔厂的模型中，求解得出需要生产 5 121 787.5 只蓝色的蜡笔，这时模型的解记为 5 121 787 或 5 121 788 都是不合理的，因为这样取整后结果可能是不可行解或非最优解。下面的小模型说明了这种情况：

```
MAX = X;
X + Y = 25.5;
X < = Y;
```

显然，该模型的最优解是 X = Y = 12.75。现在，想让 X 通过取整而获得最优解。简单地取 X 为 13 会使模型没有可行解，很明显，最优解是 X = 12，Y = 13.5。但是，对于含有大量的整型变量的大型模型来说，采用目测法求最优解是行不通的。为了解决这个问题，LINGO 提供了一

种复杂的运算规则——分支定界法。这一方法隐含地列举了所有的整数组合,以确定整数规划的最佳可行解。由于这一方法需要大量的运算时间,所以应尽量避免使用整型变量。但如果分数值没有意义,必须要使用整型变量。

### 3.2.1 普通整型变量

@GIN 函数的语法如下:

```
@GIN(variable_name);
```

式中,variable_name 是将要取整的变量名。@GIN 函数可以放在模型中输入约束条件的位置,也可以嵌入在 @FOR 函数中对全部或部分属性变量取整。下面是应用 @GIN 函数的例子:

```
!设定标量变量 X 为普通整型变量;
@GIN(X);
!设定标量变量 PRODUCE(5)为普通整型变量;
@GIN(PRODUCE(5));
!设定所有 START 属性变量为普通整型变量;
@FOR(DAYS(I):@GIN(START(I)));
```

#### 3.2.1.1 生产优化

为了说明@GIN 函数在一个完整模型中的应用,对 CompuQuick 公司模型进行一些调整,将每天生产线上 Standard 计算机的产量上限调整到 103 台。结果,生产线的约束条件为:

```
STANDARD<=103;
```

把该约束加到原始的 CompuQuick 模型中,可以得到:

```
MAX=100*STANDARD+150*TURBO;
STANDARD<=103;
TURBO<=120;
STANDARD+2*TURBO<=160;
```

解这个模型,得到它的解为:

```
Global optimal solution found at step:        0
Objective value:                          14575.00
  Variable            Value        Reduced Cost
  STANDARD          103.0000          0.0000000
  TURBO             28.50000          0.0000000
```

注意到新的 Turbo 计算机的最优生产量是 28.5 台,并不是整数值。为了确保结果切合实际,加入 @GIN 函数使变量 STANDARD 和 TURBO 都是整型的,修正后的模型为:

```
MAX=100*STANDARD+150*TURBO;
STANDARD<=103;
TURBO<=120;
```

```
STANDARD + 2 * TURBO < = 160;
@ GIN (STANDARD) ;@ GIN (TURBO) ;
```

**求解这个修正后的整数解的模型,得到如下结果:**

```
Global optimal solution found at step:        4
Objective value:                          14550.00
Branch count:                              1
  Variable           Value        Reduced Cost
  STANDARD          102.0000        -100.0000
  TURBO             29.00000        -150.0000
```

**注意到现在有一个新的统计量——分支数(Branch count),它表示在利用分支定界法求解时整型变量被强制赋给整数值的次数。如果分支数很大,表明 LINGO 会耗用很多时间来求出模型合理的整数解。同时,也要注意到 LINGO 要用 4 步确定整型模型最优解,而在这之前,只需要 0 步就可以求得连续型模型的最优解。可见,求解整数规划模型要比连续型模型困难。**

### 3.2.1.2 员工工作安排

**回想 2.6.1 节的员工安排模型的解,即在一周中每天开始工作的员工人数,尽管没有使用整型变量,还是得出了最优解,但这只是一个巧合。下面介绍员工模型。**

**在原始的员工模型中,一周中每天需要的员工数分别为 20、16、13、16、19、14 和 12。为了说明问题,把第二天的人数需求量 16 改成 12,把第三天的人数需求量 13 改成 18,模型就变为:**

```
SETS:
  DAYS/MON TUE WED THU FRI SAT SUN/:
  REQUIRED, START;
ENDSETS
DATA:
  REQUIRED = 20 12 18 16 19 14 12;
ENDDATA
MIN = @ SUM (DAYS (I) :START (I)) ;
@ FOR (DAYS (J) :
  @ SUM (DAYS (I) | I # LE # 5:
    START (@ WRAP (J - I + 1,7)))
      > = REQUIRED (J)
);
```

**当模型作这些调整后再求解,就得不到整数解了。**

```
Global optimal solution found at step:        7
Objective value:                          23.66667
  Variable           Value        Reduced Cost
  START (MON)       9.666667       0.0000000
  START (TUE)       0.0000000     -0.2980232E-07
```

```
START(WED)        3.666667        0.0000000
START(THU)        5.666667        0.0000000
START(FRI)        0.0000000       -0.2980232E-07
START(SAT)        4.666667        0.0000000
START(SUN)        0.0000000       0.3333333
```

在这个特殊的模型中,把结果取整而保持解是可行的。取整后,可能有一些天的员工数超过了需要量,但是没有一天员工人数不够,而且目标函数值变为 10 + 4 + 6 + 5 = 25 名员工。

这时,可使用整数规划建立修改后的员工模型。首先,用函数 @GIN 使变量 START 变为普通的整型变量,把以下的内容加入模型中:

```
@GIN(@START(MON));
@GIN(@START(TUE));
@GIN(@START(WED));
@GIN(@START(THU));
@GIN(@START(FRI));
@GIN(@START(SAT));
@GIN(@START(SUN));
```

当然,更简捷的方法是把 @GIN 函数放在 @FOR 函数中:

```
@FOR(DAYS(I):@GIN(START(I)));
```

以上新的语句说明,用于表示一周内每天开始工作的员工数的变量是整型的。然后,把 @FOR语句加入模型中,重新求解,得到真正的整数解:

```
Global optimal solution found at step:        6
Objective value:                             24.00000
Branch count:                                 1
  Variable           Value        Reduced Cost
  START(MON)       9.000000         1.000000
  START(TUE)       3.000000         1.000000
  START(WED)       1.000000         1.000000
  START(THU)       6.000000         1.000000
  START(FRI)       0.000000         1.000000
  START(SAT)       4.000000         1.000000
  START(SUN)       1.000000         1.000000
```

这时,目标函数的值为 24,而不是靠取整得到的 25。所以,如果用取整得到结果时,就需要多雇用一名员工,这显然是不经济的。

### 3.2.2 二进制整型变量

二进制变量也称为 0/1 变量,在只需要 0 或 1 时使用,经常被用在 Yes/No 决策的模型中。

@BIN 函数的语法为:

```
@BIN(variable_name);
```

式中，variable_name 为二进制变量的变量名。同样，@BIN 函数可以放在输入约束条件处的任何位置，也可以嵌入 @FOR 函数语句中。很容易将所有或部分代表属性值的变量设置成二进制变量。下面是 @BIN 函数的一些例子：

```
!把标量变量 X 设置成一个二进制变量;
@BIN(X);
!把变量 INCLUDE(4)设置成一个二进制变量;
@BIN(INCLUDE(4));
!把所有代表 INCLUDE 属性的变量设置成二进制变量;
@FOR(ITEMS:@BIN(INCLUDE));
```

#### 3.2.2.1 二进制变量的例子——背包问题

背包问题是使用二进制变量的经典问题。在这一问题中，有一批物品将要放入背包中，但是背包的容量有限，不可能装下所有的东西。每一样物品都有一定价值或功用，而且这些属性与其是否将被放入背包直接相关。所以，这个问题可表述为：在不超过背包容量的情况下，使装入背包的物品的总价值最大。

背包问题可以应用到很多问题的求解过程中，例如车辆的装载量、预算或决策的制定等。

1. 一个简单的背包问题

假设去野餐，需要列一个携带物品的清单。每一样物品都有重量，而背包能承受的最大重量是 15 kg。物品分为 10 种权重，权重代表了这一物品对于野餐的重要性。各物品的信息列于表 3-1 中。

表 3-1 物品重量和权重

| 物品 | 重量 (kg) | 权重 | 物品 | 重量 (kg) | 权重 |
|---|---|---|---|---|---|
| Ant Repellent | 1 | 2 | Brownies | 3 | 10 |
| Beer | 3 | 9 | Frisbee | 1 | 6 |
| Blanket | 4 | 3 | Salad | 5 | 4 |
| Bratwurst | 3 | 8 | Watermelon | 10 | 10 |

2. 建立模型

在这个模型中只有一个集合，即可能装入背包的物品的集合。可以在集合域建立这一基本集合：

```
SETS:
  ITEMS/ANT_REPEL,BEER,BLANKET,
    BRATWURST,BROWNIES,FRISBEE,SALAD,
      WATERMELON/:
        INCLUDE,WEIGHT,RATING;
ENDSETS
```

在这个集合中有三个相关联的属性 INCLUDE、WEIGHT 和 RATING。INCLUDE 是二进制变量,用于表示某个物品是否放入背包中;WEIGHT 用于表示每个物品的重量;RATING 用于表示每个物品的权重。

另外,需要在数据域中输入每种物品的重量和权重:

```
DATA:
  WEIGHT RATING =
    1      2
    3      9
    4      3
    3      8
    3      10
    1      6
    5      4
    10     10;
  KNAPSACK_CAPACITY = 15;
ENDDATA
```

注意:背包的容量也包含在数据域中。在建立模型时,将数据和约束条件分离是非常好的习惯。

在给出所有的集合和数据后,开始建立目标函数,即力争使背包中物品权重值的和最大。当某个物品被选中时,它的 INCLUDE(i)值为 1,否则为 0。因此,作 RATING 和 INCLUDE 值的内积,就可得到所有放入背包的物品的权重之和。下面是 LINGO 语句:

```
MAX = @SUM(ITEMS:RATING * INCLUDE);
```

注意:没有在@SUM 函数中指定集合的索引变量,因为函数中所有的属性值(RATING 和 INCLUDE)都是在带有索引的集合(ITEMS)中定义的。

下一步是输入约束条件。这个模型中只有一个约束条件,即携带的物品不能超过背包的容量。类似于目标函数,作 WEIGHT 和 INCLUDE 属性的内积可得到放入背包中的物品的总重量。这一求和值一定小于或等于背包的容量,用 LINGO 语句表达为:

```
@SUM(ITEMS:WEIGHT * INCLUDE) < =
  KNAPSACK_CAPACITY;
```

最后,通过添加以下函数使变量 INCLUDE 成为二进制变量:

```
@BIN(INCLUDE(@INDEX(ANT_REPEL)));
@BIN(INCLUDE(@INDEX(BEER)));
@BIN(INCLUDE(@INDEX(BLANKET)));
@BIN(INCLUDE(@INDEX(BRATWURST)));
@BIN(INCLUDE(@INDEX(BROWNIES)));
@BIN(INCLUDE(@INDEX(FRISBEE)));
@BIN(INCLUDE(@INDEX(SALAD)));
@BIN(INCLUDE(@INDEX(WATERMELON)));
```

这里，@INDEX 函数返回基本集合中相应元素的索引值。另外一种更高效且与数据相互独立的方法是把 @BIN 函数嵌入 @FOR 函数中，即：

```
@FOR(ITEMS:@BIN(INCLUDE));
```

3. 求解结果

整个背包问题的模型和部分模型的解如下（模型文件可以在 LINGO 主目录下的子目录 SAMPLES 中找到，名为 KNAPSACK）：

```
SETS:
  ITEMS/ANT_REPEL,BEER,BLANKET,
  BRATWURST,BROWNIES,FRISBEE,SALAD,
    WATERMELON/:
      INCLUDE,WEIGHT,RATING;
ENDSETS
DATA:
  WEIGHT RATING=
    1      2
    3      9
    4      3
    3      8
    3      10
    1      6
    5      4
   10      10;
  KNAPSACK_CAPACITY=15;
ENDDATA
MAX=@SUM(ITEMS:RATING*INCLUDE);
@SUM(ITEMS:WEIGHT*INCLUDE)<=
  KNAPSACK_CAPACITY;
@FOR(ITEMS:@BIN(INCLUDE));
Global optimal solution found at step:             10
Objective value:                              38.00000
Branch count:                                 0
    Variable                           Value
  INCLUDE(ANT_REPEL)                1.000000
  INCLUDE(BEER)                     1.000000
  INCLUDE(BLANKET)                  1.000000
  INCLUDE(BRATWURST)                1.000000
  INCLUDE(BROWNIES)                 1.000000
  INCLUDE(FRISBEE)                  1.000000
  INCLUDE(SALAD)                    0.000000
  INCLUDE(WATERMELON)               0.000000
```

背包中正好装了 15 kg 的物品，带了除 Salad 和 Watermelon 外的所有物品，午餐中的 Beer、Bratwurst、Brownies 可能并不是健康的食品，但是却可以使旅游者快乐。

4. 扩充——用模型表达逻辑“或”

二进制变量对于用模型表达逻辑“或”很有用。例如，为了健康，假设医生检查野餐菜单后建议旅游者带上沙拉或西瓜中的一种，可以在模型中加入一个约束条件：

```
INCLUDE(@INDEX(SALAD)) + INCLUDE(@INDEX(WATERMELON)) >= 1;
```

这种约束条件的表达形式并不完善，因为它们与数据不是相互独立的。假设野餐物品的列表要改动，则约束条件也要变化。一个表达完善的模型在修改数据时应不需要改变约束条件。以下的模型说明了一种在数据独立的情况下加入医生建议的约束条件（原始模型中增加的部分用黑体列出）：

```
SETS:
  ITEMS/ANT_REPEL,BEER,BLANKET,
    BRATWURST,BROWNIES,FRISBEE,SALAD,
      WATERMELON/:
        INCLUDE,WEIGHT,RATING;
  MUST_EAT_ONE(ITEMS)
    /SALAD,WATERMELON/;
ENDSETS
DATA:
  WEIGHT RATING=
    1      2
    3      9
    4      3
    3      8
    3      10
    1      6
    5      4
   10      10;
  KNAPSACK_CAPACITY=15;
ENDDATA
MAX=@SUM(ITEMS:RATING*INCLUDE);
@SUM(ITEMS:WEIGHT*INCLUDE)<=
  KNAPSACK_CAPACITY;
@FOR(ITEMS:@BIN(INCLUDE));
@SUM(MUST_EAT_ONE(I):INCLUDE(I))>=1;
```

从原始的背包模型中衍生出一个叫做 MUST_EAT_ONE 的集合，用直接列表表示必须带的物品。在模型的最后加入约束条件，使结果中至少包含一种“必须吃”的物品。修改后的模型的结果为：

```
Global optimal solution found at step:              11
```

```
Objective value:                              37.00000
Branch count:                                     0
  Variable                                    Value
INCLUDE(ANT_REPEL)                          0.000000
INCLUDE(BEER)                               1.000000
INCLUDE(BLANKET)                            0.000000
INCLUDE(BRATWURST)                          1.000000
INCLUDE(BROWNIES)                           1.000000
INCLUDE(FRISBEE)                            1.000000
INCLUDE(SALAD)                              1.000000
INCLUDE(WATERMELON)                         0.000000
```

最后，去掉了 Ant Repellent 和 Blanket，代替它们的是 Salad。

### 3.2.2.2 二进制整型变量举例——考虑固定成本的生产安排

在许多情况下，特定的活动都与固定成本相联系。例如开设工厂、生产产品、股票市场上的委托交易和重组生产线等。在接下来的例子中，将建立一个与 CompuQuick 模型类似的产品生产模型。在这个例子中，生产产品需要一定的基建投资。这项投资与产品的产出水平无关，无论生产多少产品，都需要一个固定资产投资。

1. 一个固定成本模型

假设一个生产飞机的工厂需要决定六种型号飞机的生产组合。准备生产的六种型号包括 Rocket、Meteor、Streak、Comet、Jet 和 Biplane。每种飞机都有已知的利润值，而且一定时期内生产飞机需要一定的固定资产投资。利润值和固定资产投资值在表 3-2 中给出。

表 3-2 各型号飞机的利润和固定资产投资

| 型号 | 利润 ($10^5$ 元) | 固定资产投资 ($10^5$ 元) | 型号 | 利润 ($10^5$ 元) | 固定资产投资 ($10^5$ 元) |
|---|---|---|---|---|---|
| Rocket | 30 | 35 | Comet | 26 | 70 |
| Meteor | 45 | 20 | Jet | 24 | 75 |
| Streak | 24 | 60 | Biplane | 30 | 30 |

生产每种飞机都要用到六种原材料，即钢铁、铜、塑料、橡胶、玻璃和油漆。生产每种飞机需要的原材料量和总共可以提供的原材料量如表 3-3 所示。

表 3-3 资源表

| | Rocket | Meteor | Streak | Comet | Jet | Biplane | 可利用资源 (吨) |
|---|---|---|---|---|---|---|---|
| 钢铁 (吨) | 1 | 4 | 0 | 4 | 2 | 0 | 800 |
| 铜 (吨) | 4 | 5 | 3 | 0 | 1 | 0 | 1 160 |
| 塑料 (吨) | 0 | 3 | 8 | 0 | 1 | 0 | 1 780 |
| 橡胶 (吨) | 2 | 0 | 1 | 2 | 1 | 5 | 1 050 |
| 玻璃 (吨) | 2 | 4 | 2 | 2 | 2 | 4 | 1 360 |
| 油漆 (吨) | 1 | 4 | 1 | 4 | 3 | 4 | 1 240 |

这个模型是为了确定在不超过原材料可利用量的条件下的生产安排方案,以使净利润(总利润 - 固定资产投资)最大。可以发现,生产 Meteor 型号的飞机具有最高的利润值和最低的固定资产投资,这不是只要生产 Meteor 而不生产别的机型了吗?计算后的结果可能不是这样。

2.建立模型

在这个模型中,需要两个基本集合,一个代表飞机型号,一个代表原材料。模型如下:

```
PLANES/ROCKET,METEOR,STREAK,
  COMET,JET,BIPLANE/:
    PROFIT,SETUP,QUANTITY,BUILD;
RESOURCES/STEEL,COPPER,PLASTIC,
  RUBBER,GLASS,PAINT/:AVAILABLE;
```

在 PLANES 集合中加入了以下四个属性:

PROFIT——每种型号飞机的净利润值;

SETUP——开始生产前的基建费用;

QUANTITY——飞机生产数量;

BUILD——二进制变量,1 代表生产某型号飞机,0 代表不生产。

集合 RESOURCES 的 AVAILABLE 属性用来表示每种资源的可利用量。

还需要通过 RESOURCES 集合与 PLANES 集合衍生出一个稠密集合,并且定义 USAGE 属性表示每种飞机对原料的使用情况。这一集合定义为:

```
RXP(RESOURCES,PLANES):USAGE;
```

在数据域中,必须进行初始化的数据属性为 PROFIT、SETUP、AVAILABLE 和 USAGE:

```
DATA:
  PROFIT SETUP=
    30    35
    45    20
    24    60
    26    70
    24    75
    30    30;
  AVAILABLE=
    800 1160 1780 1050 1360 1240;
  USAGE=  1 4 0 4 2 0
          4 5 3 0 1 0
          0 3 8 0 1 0
          2 0 1 2 1 5
          2 4 2 2 2 4
          1 4 1 4 3 4;
ENDDATA
```

完成集合域和数据域后,就可以建立模型的目标函数。该模型的目标是使净产值最大化。

明确地说,就是生产的每种飞机的数量与其利润之积的和,减去它们的基建费用乘以二进制变量 BUILD 的积的和。用 LINGO 语句表达的目标函数为:

```
MAX = @SUM(PLANES:
  PROFIT * QUANTITY - SETUP * BUILD);
```

因为所有的属性都是在具有索引的集合 PLANES 中定义的,所以可以不使用集合的索引变量。

对于第一组约束条件,希望原料用量不超过提供值,即对于第 I 种资源,每种飞机 J 的使用量乘以飞机 J 的生产数量要小于或等于资源 I 的可提供量:

```
@FOR(RESOURCES(I):
  @SUM(PLANES(J):
    USAGE(I,J) * QUANTITY(J)) < =
      AVAILABLE(I)
);
```

另一组约束条件并不很直观,用二进制变量 BUILD 表示某种飞机是否生产,并且当生产飞机 I 的量为非零值时,把 BUILD(I) 设置成为 1。以下语句实现这种功能:

```
@FOR(PLANES:
  QUANTITY < = 400 * BUILD;
    @BIN(BUILD)
);
```

对于 0/1 变量 BUILD,只要变量 QUANTITY 为非零值,对应的 BUILD 唯一可行值为 1。因为这种形式的约束强制把 0/1 变量取到合适的值,所以这些约束有时被称作强制约束。

BUILD 的系数取 400 是因为从数据中可以看出,生产的每种型号的飞机都不可能多于 400 架。这种方式下使用的系数有时被称为 Big M 系数。为了保证求解效率,应尽可能减小 Big M 的值。事实上,像这样在模型的约束中加入系数对于模型的求解是不利的。正如所讨论的,最好使模型的约束条件与模型的数据保持独立,以方便模型的维护,所以一个数据独立性更强的模型应有专门的计算式计算 Big M 的值。

现在一个值得考究的问题是:当生产这种飞机时,用一种方式把 BUILD 值取为 1,但是当不生产这种飞机时,怎样把 BUILD 设置成 0?在目标函数中,BUILD 被设置成负系数(从目标函数中减去 BUILD 乘以 SETUP 的值)来保证实现这种赋值。具体地讲,如果不生产此种飞机,而它相应的变量 BUILD 值却被设置为 1,这时只要将 BUILD 设置成 0,就能使目标函数得到改善,所以当飞机不生产时变量 BUILD 的值肯定为 0。

作为一个重要的性质,可以利用下面的语句把 QUANTITY 变量设置成普通整型变量:

```
@FOR(PLANES:
  @GIN(QUANTITY)
);
```

3. 求解结果

模型文件可以在 PRODMIX 文件中找到,整个模型的表达式的一部分求解结果如下:

```
SETS:
  PLANES/ROCKET,METEOR,STREAK,
    COMET,JET,BIPLANE/:
      PROFIT,SETUP,QUANTITY,BUILD;
  RESOURCES/STEEL,COPPER,PLASTIC,
    RUBBER,GLASS,PAINT/:AVAILABLE;
  RXP(RESOURCES,PLANES):USAGE;
ENDSETS
DATA:
  PROFIT SETUP=
    30    35
    45    20
    24    60
    26    70
    24    75
    30    30;
  AVAILABLE=
    800 1160 1780 1050 1360 1240;
  USAGE=  1 4 0 4 2 0
          4 5 3 0 1 0
          0 3 8 0 1 0
          2 0 1 2 1 5
          2 4 2 2 2 4
          1 4 1 4 3 4;
ENDDATA
MAX=@SUM(PLANES:
  PROFIT*QUANTITY-SETUP*BUILD);
@FOR(RESOURCES(I):
  @SUM(PLANES(J):
    USAGE(I,J)*QUANTITY(J))<=
      AVAILABLE(I)
);
@FOR(PLANES:
  QUANTITY<=400*BUILD;
    @BIN(BUILD)
);
@FOR(PLANES:
  @GIN(QUANTITY)
);
Global optimal solution found at step:       73
Objective value:                     14764.00
Branch count:                            17
```

```
  Variable                  Value
QUANTITY(ROCKET)          96.00000
QUANTITY(METEOR)          0.000000
QUANTITY(STREAK)          195.0000
QUANTITY(COMET)           0.000000
QUANTITY(JET)             191.0000
QUANTITY(BIPLANE)         94.00000
BUILD(ROCKET)             1.000000
BUILD(METEOR)             0.000000
BUILD(STREAK)             1.000000
BUILD(COMET)              0.000000
BUILD(JET)                1.000000
BUILD(BIPLANE)            1.000000
```

可以发现,结果中并没有利润最大的 Meteor,而是生产 Rocket、Streak 和 Jet 三种型号的飞机。

## 3.3 自由变量

在默认状态下,LINGO 中变量的下限是 0,上限是正无穷大。但是,@FREE 函数可以使变量取负值,使它不再受下限 0 的限制,其语法为:

```
@FREE(variable_name);
```

其中,variable_name 是希望成为自由变量的变量名。同样,@FREE 函数可以放在模型中输入约束条件的任意位置,也可以嵌入 @FOR 函数语句中,实现把所有的变量或选定的某一属性变量设置成为自由变量。下面是 @FREE 函数的一些例子:

```
!把变量 X 设置成自由变量;
@FREE(X);
!把变量 QUANTITY(4)设置成自由变量;
@FREE(QUANTITY(4));
!把所有代表 INCLUDE 属性的变量设置成自由变量;
@FOR(ITEMS:@FREE(QUANTITY));
```

### 3.3.1 自由变量举例——预测

假设有一个名为 Shack4Shades 的零售公司的库存管理员,其业务是专门研究户外活动爱好者需要的太阳镜的销售情况。所以,需要建立一个模型用于预测太阳镜在下一个季度的销售情况,以确定存货的数量。图 3-1 表示前 8 个季度该公司的销售情况。

可见,在图中将销售情况理论化成了线性趋势线,该趋势线伴随着相当大的季节变化。销售的一个高峰是夏天,因为很多人要去海滩;另一个高峰是冬季,因为很多人去滑雪。根据图 3-1 可以写出以下的销售预测的函数式:

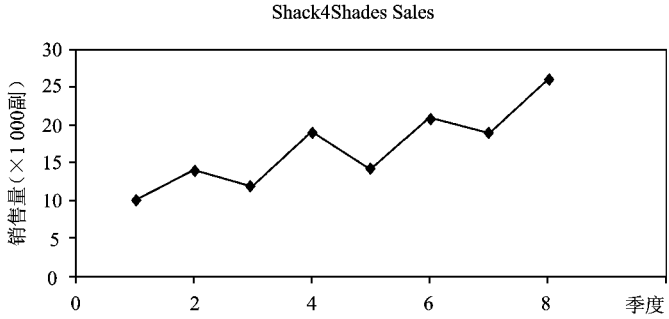



**图 3-1　销售趋势图**

$$Predicted\_Sales(t) = Seasonal\_Factor(t) \times (Base + Trend \times t)$$

式中，Predicted _ Sales (t) 为第 t 季度的预测销售值，1 000 副；Seasonal _ Factor (t) 为季度因子，代表季节性变化；Base 为设定的线性函数在 y 轴上的截距；Trend 为趋势线的斜率。

根据前 8 个季度的销售情况，可以建立一个 LINGO 模型来估计上述方程的 6 个参数，使得预测值与历史销售实际值的差值的平方和最小。

## 3.3.2　建立模型

在这个模型中需要两个基本集合。第一个集合有 8 个元素，代表拥有历史数据的 8 个季度；第二个集合有 4 个元素，代表一年中的 4 个季度，用来定义季节影响因素。下面是这两个集合的定义：

```
SETS:
  PERIODS/1..8/:OBSERVED,PREDICT,
    ERROR;
  QUARTERS/1..4/:SEASFAC;
ENDSETS
```

PERIODS 集合有 OBSERVED、PREDICT 和 ERROR 三个属性，分别代表已知销售值、预测销售值和预测误差。预测误差表示为预测销售值减去已知销售值。集合 QUARTERS 中的 SEASFAC 属性代表季度因子，需要用 LINGO 计算得到。

然后，在模型中加入数据域，利用历史数据为 OBSERVED 属性赋值：

```
DATA:
  OBSERVED = 10 14 12 19 14 21 19 26;
ENDDATA
```

接下来，为计算误差项增加一个表达式。正如上文提到的，误差代表预测值与实际值的差异，可以在 LINGO 中表达为：

```
@FOR(PERIODS:ERROR = PREDICT - OBSERVED);
```

模型的目的是使误差的平方和最小：

```
MIN = @SUM(PERIODS:ERROR^2);
```

之所以选择误差值的平方和作为标准，是因为这样可以使相对大的误差值占的比重更大一些。当然，还可以采用误差的绝对值的和作为依据，这样就使各个误差值具有相同的比重。

为了计算出误差值，还需要计算出预测值。利用理想化的公式计算出的预测值如下：

```
@FOR(PERIODS(P):PREDICT(P)=SEASFAC(@WRAP(P,4))*(BASE+P*TREND));
```

式中使用了 @WRAP 函数，可以在超过 4 个时期的时间轴上应用 4 个季度因子，并且可以通过添加约束条件实现季度因子的平均值为 1。添加的约束条件如下：

```
@SUM(QUARTERS:SEASFAC)=4;
```

最后，因为误差值可以是正数也可以是负数，所以要使用 @FREE 函数使误差值可以取到负值：

```
@FOR(PERIODS:@FREE(ERROR));
```

### 3.3.3 结果

整个模型表达式和部分计算结果如下：

```
SETS:
  PERIODS/1..8/:OBSERVED,PREDICT,
  ERROR;
  QUARTERS/1..4/:SEASFAC;
ENDSETS
DATA:
  OBSERVED=10 14 12 19 14 21 19 26;
ENDDATA
MIN=@SUM(PERIODS:ERROR^2);
@FOR(PERIODS:ERROR=
  PREDICT-OBSERVED);
@FOR(PERIODS(P):PREDICT(P)=
  SEASFAC(@WRAP(P,4))
    *(BASE+P*TREND));
@SUM(QUARTERS:SEASFAC)=4;
@FOR(PERIODS:@FREE(ERROR));
Optimal solution found at step:        27
Objective value:                  1.822561
    Variable           Value
  BASE               9.718875
  TREND              1.553017
  PREDICT(1)         9.311819
  PREDICT(2)         14.10136
  PREDICT(3)         12.85212
  PREDICT(4)         18.80620
```

| | |
|---|---|
| PREDICT (5) | 14.44367 |
| PREDICT (6) | 20.93171 |
| PREDICT (7) | 18.40496 |
| PREDICT (8) | 26.13944 |
| ERROR (1) | -0.6881806 |
| ERROR (2) | 0.1013619 |
| ERROR (3) | 0.8521245 |
| ERROR (4) | -0.1938018 |
| ERROR (5) | 0.4436698 |
| ERROR (6) | -0.6828707E-01 |
| ERROR (7) | -0.5950383 |
| ERROR (8) | 0.1394362 |
| SEASFAC (1) | 0.8261097 |
| SEASFAC (2) | 1.099529 |
| SEASFAC (3) | 0.8938788 |
| SEASFAC (4) | 1.180482 |

从上面的求解结果看，4 个季度因子分别为 0.826、1.01、0.894 和 1.18。其中，春季的季度因子为 0.826，也就是说春天卖到平均水平的 82.6%。斜率 1.55 表示除去要考虑的季节因素外，太阳镜的销售以每个季度 1 550 副的数量增长。预测函数得出的结果与历史资料非常吻合，如图 3-2 所示。

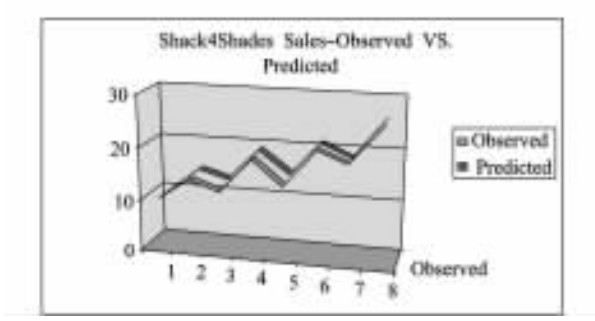



图 3-2　预测结果图

利用得到的预测式，可以预测第 9 季度的销售值：

$$\begin{aligned}\text{Predicted\_Sales}(9) &= \text{Seasonal\_Factor}(1) * (\text{Base} + \text{Trend} * 9)\\ &= 0.826 * (9.72 + 1.55 * 9)\\ &= 19.55\end{aligned}$$

根据这一结果，存货应该保持足够的水平，以保证将要卖出的 19 550 副眼镜。

## 3.4　限界变量

区别于 @FREE 函数将变量取值的上限和下限分别定义为正无穷大和负无穷大，@BND 函数可以任意指定一个变量的上限和下限。换句话说，@BND 函数限定变量在一个固定的区

间内取值。其语法为：

```
@BND(lower_bound,variable_name,upper_bound);
```

式中，variable_name 为变量名。该变量的取值上限为 upper_bound，下限为 lower_bound。上限和下限都必须是数值或已经在数据域中赋值的变量。@BND 函数可以放在模型中输入约束条件处的任何地方，也可以包含在 @FOR 循环函数中。

可以用约束条件代替 @BND 函数。但是，从优化建模的角度看，@BND 函数是表示变量简单取值范围的一种非常有效的方法，因为使用 @BND 函数表达变量的取值范围比使用约束更能显著地提高大型模型的求解速度，而且 @BND 函数不计入 LINGO 的约束条件的数目中，这样有助于避开某些版本的 LINGO 对约束条件数目的限制。因此，在通常情况下，应尽量使用该函数来代替约束条件。下面是 @BND 函数的一些例子：

```
!把变量 X 限定在区间 [-1,1]内;
@BND(-1,X,1);
!限定变量 QUANTITY(4)落入 100 到 200 之间;
@BND(100,QUANTITY(4),200);
!限定 Q 中所有变量的取值范围为 10 到 20;
@FOR(ITEMS:@BND(10,Q,20));
!把属性 Q 的所有变量值限定在 QL 和 QU 之间(QL 和 QU 必须事先在数据域中赋值);
@FOR(ITEMS:@BND(QL,Q,QU));
```

# 第 4 章　数据域和初始化域

一般情况下,在使用 LINGO 求解模型之前,必须先明确集合的元素并给集合的一些属性赋值。为此,LINGO 为用户提供了两个可选域,即数据域和初始化域。其中,数据域用来输入集合元素和数据的值,初始化域则用于设置决策变量的初始值。

## 4.1　模型的数据域

### 4.1.1　数据域简介

数据域可以使数据与模型的其他部分分离。这一重要作用使模型易于维护,也利于改变模型的维数。

数据域以关键字"DATA:"开始,以关键字"ENDDATA"结束。在数据域中可以将在集合域中建立的集合元素和属性值初始化,语法如下:

```
object list = value list;
```

object list (对象列表) 包含将要初始化的属性名或集合名,并且可以选择用逗号将各个对象分开。一个列表中只能有一个集合名称,但是可以有很多个属性。如果 object list 中有多于一个的属性名,则这些属性必须是在同一个集合中定义的。如果 object list 中有一个集合名称,则 object list 中所有的属性都必须在该集合中被定义。

value list (值列表) 包含将要给 object list 中的成员赋的值,同样可以用逗号将这些值分开,例如下面的模型:

```
SETS:
  SET1/A,B,C/:X,Y;
ENDSETS
DATA:
  X = 1,2,3;
  Y = 4,5,6;
ENDDATA
```

在集合 SET1 中定义了两个属性 X 和 Y。X 的值取 1、2、3;Y 的值取 4、5、6。也可以使用如下混合赋值语句:

```
SETS:
  SET1/A,B,C/:X,Y;
ENDSETS
DATA:
```

```
  X,Y = 1,4,
        2,5,
        3,6;
ENDDATA
```

在这个模型中,X 的值是 1、2、3。如前所述,当 LINGO 读取一个赋值语句中的数据列表时,会把数据列表中头 n 个值分别赋给属性列表中的 n 个属性的第一个位置,把数据列表中第二组的 n 个值分别赋给属性列表中的 n 个属性的第二个位置,依此类推。换句话说,LINGO 赋值时是按列而不是按行进行的。

正如前面提到的,也可以在数据域中将集合元素初始化。利用把集合元素从集合域移到数据域这一方式来修改模型,可以得到:

```
SETS:
  SET1 :X,Y;
ENDSETS
DATA:
  SET1,X,Y = A 1 4
             B 2 5
             C 3 6;
ENDDATA
```

以上这种方法或许是最佳的,因为模型中所有的数据(属性值和集合元素)均独立于数据域中。

### 4.1.2 参数

在 LINGO 中,还可以在数据域中对变量进行初始化。当一个数值变量在数据域中取一个定值时,称之为参数。

例如,假设在模型中把利率 8.5%作为一个参数,则可以在数据域中按下面的方式把利率作为一个参数输入:

```
DATA:
  INTEREST_RATE = .085;
ENDDATA
```

如同集合属性,LINGO 可以在一个语句中为多个参数赋值。假设在模型中增添通货膨胀率,可以在同一个赋值语句中同时定义利率和通货膨胀率:

```
DATA:
  INTEREST_RATE,INFLATION_RATE = .085,.03;
ENDDATA
```

### 4.1.3 估值分析

在某些情况下,无法确定给模型中的数据输入什么值是最合适的。例如,假设模型用通货

膨胀率作为参数，虽然不能确定未来的通货膨胀率的确定值，但是可以知道其值会落入 2% ~ 6%之中。因此，想确定的是在这个取值范围内运行这个模型时，通货膨胀率对模型有多大影响。LINGO 有一个功能可以方便地实现这一目的，把这一分析过程称为估值分析（What If Analysis）。可为估值分析设置一个参数，并且输入“?”代替相应的参数值，例如：

```
DATA:
  INFLATION_RATE = ?;
ENDDATA
```

每次求解模型时，都会出现图 4-1 所示对话框，提示为参数 INFLATION_RATE 输入一个数值。

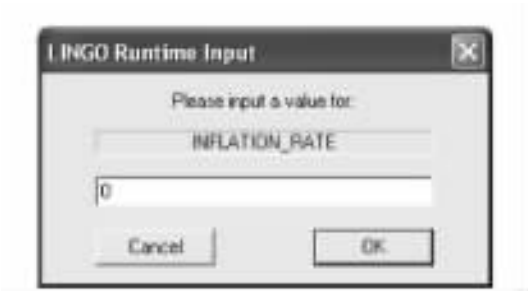



图 4-1　参数输入对话框

在此对话框中，输入通货膨胀率的值再单击 OK 按钮，LINGO 就会把输入值赋给 INFLATION_RATE，然后求解这个模型。

在其他操作系统下，LINGO 将在屏幕上给出提示，询问是否为 INFLATION_RATE 赋值，输入数值后按 Enter 键，同样可以求解模型。

除了参数之外，还可以把属性中的单独元素赋值为问号进行估值分析。

### 4.1.4　把属性初始化为同一个固定值

假设要把所有元素的同一属性初始化为同一个固定值，可以在赋值语句的右边输入一个固定值实现。在执行估值分析时，如果将属性值初始化为一个问号，每次求解模型时 LINGO 都会提醒为属性中的所有元素赋值。例如：

```
SETS:
  DAYS/MO,TU,WE,TH,FR,SA,SU/:
  NEEDS;
ENDSETS
DATA:
  NEEDS = 20;
ENDDATA
```

LINGO 将属性 NEEDS 的所有元素初始化为 20。

如果赋值语句的左边存在多个属性，就要在赋值语句的右边为左边的每一个属性输入一个值。例如：

```
SETS:
  DAYS/MO,TU,WE,TH,FR,SA,SU/:
  NEEDS,COST;
ENDSETS
DATA:
  NEEDS,COST = 20,100;
ENDDATA
```

LINGO 将属性 NEEDS 的 7 个元素初始化为 20,而将属性 COST 的 7 个元素初始化为 100。

### 4.1.5 在数据域中省略值

当不想为特定的元素赋固定值时,可以在赋值语句中省略它们的值。例如,有一个工厂要做下一个 5 年的生产计划:

```
SETS:
  YEARS/1..5/:CAPACITY;
ENDSETS
DATA:
  CAPACITY = 34,34,,,;
ENDDATA
```

模型中,把前两年的 CAPACITY 值设置为 34,但是省略最后 3 年的值。这样,LINGO 认为最后 3 年的 CAPACITY 可以自由取值。

注意 在省略值时必须使用逗号,如果不使用逗号,LINGO 会认为 CAPACITY 的赋值不正确,会显示错误信息。

## 4.2 模型的初始化域

初始化域是 LINGO 提供的一个可选部分,其中的初始化语句与数据域中的赋值语句类似。在初始化域中输入的数值将作为 LINGO 求解的起始点。与数据域不同的是,在数据域中赋值的变量的值不能再改变,而初始化域中的变量的值可以随着求解的过程而改变。

值得注意的是,初始化域中的起始点只能用于非线性模型中,在纯线性模型中没有任何作用。

例如,定义一个股票的集合,可能知道每种股票的价格,但是买入和卖出股票的数量是未知的。因此,一般在数据域中将价格属性的值初始化,而如果知道买入和卖出属性的大概值,可以在 LINGO 的初始化域中给出这些信息。LINGO 利用这些值作为起始点寻找最优解,如果起始点非常接近最优解,可以节省大量求解模型的时间。

初始化域以关键字"INIT:"开始,以关键字"ENDINIT"结束。初始化域中的语法规则与数据域中的语句的语法规则相同。与数据域类似,在赋值语句左边可以有多个属性,可以同时把它们初始化为固定值,也可以省略掉某些属性的值,还可以使用问号让 LINGO 在求解模型时提醒将一些变量初始化。

下面的模型说明了一个好的起始点怎样减少求解时间：

```
Y < = @ LOG (X) ;
X^2 + Y^2 < = 1 ;
```

函数@LOG(X)是取 X 的自然对数。这个模型仅有一个可行点(X,Y) = (1,0)。如果不使用初始化域解这个模型,将得到下面的结果：

```
Feasible solution found at step:      12
  Variable           Value
  Y               0.5721349E-03
  X               1.000419
```

可见,求解过程中进行了 12 次迭代。现在,增加一个初始化域把 X 和 Y 初始化为接近可行解的值：

```
INIT:
  X = .999 ;
  Y = .002 ;
ENDINIT
Y < = @ LOG (X) ;
X^2 + Y^2 < = 1 ;
```

求解上面的模型,将得到下面的结果：

```
Feasible solution found at step:       3
  Variable           Value
  X               0.9999995
  Y               0.0000000
```

可见,利用初始化域仅用了 3 次迭代就完成了上面用 12 次迭代完成的求解过程。

本章中介绍了数据域和初始化域的基本属性,以后的章节将介绍怎样在域中增加指令使其能与外部的文件、电子表格和数据库链接起来。

# 第 5 章　窗口命令

这一章将讨论 Windows 操作系统下的 LINGO 下拉菜单命令的使用方法。如果用户使用的是 LINGO 的命令行版本，可以参阅第 6 章。

## 5.1　窗口命令的使用

在 Windows 操作系统下，有三种访问命令的方法，分别是从下拉菜单中选择、单击工具栏按钮和输入相应的键盘命令（即命令的快捷键）。

### 5.1.1　菜单

在 Windows 的版本中，LINGO 将命令分组放在以下五个主菜单中：

1）File；

2）Edit；

3）LINGO；

4）Window；

5）Help。

File 菜单主要包括处理输入和输出的命令；Edit 菜单含有能够对当前窗口中文本进行编辑的命令；LINGO 菜单含有求解模型和生成结果报告的命令；Window 菜单中的命令用于处理多重窗口；Help 菜单提供了访问帮助文件的命令。

### 5.1.2　工具栏

在默认情况下，工具栏位于屏幕的顶部。工具栏如图 5-1 所示。

图 5-1　LINGO 工具栏

LINGO 的工具栏是浮动式的，可以用鼠标把它拖到屏幕上的任何位置，还可以通过 LINGO | Options 对话框中 Interface 页面上的 Toolbar 复选框选择是否隐藏工具栏。

工具栏上的每一个按钮都对应一个菜单命令，但并不是每一个菜单命令都在工具栏上有相应的按钮。

LINGO 为每一个按钮提供了“提示”信息。当用户将鼠标移动到按钮上时，一个关于按钮功能的简短描述就会出现在窗口上和屏幕底部的状态栏上。

图 5-2 是各个工具栏按钮及其相应的菜单命令。


图 5-2　工具栏按钮与菜单命令

### 5.1.3　快捷键

除了可以从菜单和工具栏访问命令外，还可以通过快捷键对 LINGO 命令进行访问，每一个菜单命令的相应快捷键都在其旁边列出。

## 5.2　主要窗口命令

在这一节中，将给出 Windows 版本下 LINGO 中的命令列表。如前所述，这些命令被分成 File、Edit、LINGO、Window 和 Help 五个主菜单，每个主菜单的命令分别如表 5-1、5-2、5-3、5-4、5-5 所示，各个命令的详细内容将在下面的章节中介绍。

表 5-1　File 菜单的主要命令

| 命令 | 功能 |
|---|---|
| New | 新建一个模型窗口 |
| Open | 打开一个已保存在磁盘里的模型 |
| Save | 将当前窗口中的内容保存到磁盘里 |
| Save As | 将当前窗口中的内容以一个新的名字保存 |
| Close | 关闭当前窗口 |
| Print | 打印当前窗口中的内容 |

续表

| 命令 | 功能 |
| --- | --- |
| Print Setup | 配置打印机 |
| Print Preview | 将当前窗口的内容按打印时的样式显示 |
| Log Output | 打开一个日志文件,保存输出到命令窗口的日志 |
| Take Commands | 运行包含在文件中的命令脚本 |
| Import LINDO File | 将一个 LINDO 文本文件转换成为一个 LINGO 模型 |
| Export File | 以 MPS 或 MPI 文件格式输出模型 |
| License | 提示输入新的许可证密码以升级 LINGO 系统 |
| Database User Info | 提示输入用户标识符和密码以通过@ODBC()函数访问数据库 |
| Exit | 退出 LINGO |

表 5-2　Edit 菜单的主要命令

| 命令 | 功能 |
| --- | --- |
| Undo | 撤销最近一次更改 |
| Redo | 恢复最后一次撤销的命令 |
| Cut | 将文档中当前选择的内容剪切到剪贴板 |
| Copy | 将当前选择的内容复制到剪贴板 |
| Paste | 将剪贴板中的内容粘贴到文档中 |
| Paste Special | 将剪贴板中的内容粘贴到文档中,并允许选择如何粘贴目标 |
| Select All | 选择当前窗口中的全部内容 |
| Find | 在文档中查找指定的文本字符串 |
| Find Next | 重复查找最后指定的字符串 |
| Replace | 用新的字符串代替指定的文本字符串 |
| Go To Line | 移动光标到指定的行号 |
| Match Parenthesis | 查找与已选括号匹配的另一半括号 |
| Paste Function | 粘贴已选择的 LINGO 函数的模板 |
| Select Font | 指定已选文本块的字体 |
| Insert New Object | 将一个 OLE 对象嵌入文档 |
| Links | 控制文档中链接的外部对象 |
| Object Properties | 指定选择的嵌入对象的属性 |

表 5-3　LINGO 菜单的主要命令

| 命令 | 功能 |
| --- | --- |
| Solve | 求解当前窗口中的模型 |
| Solution | 生成一个当前模型的结果报告 |
| Range | 为当前窗口生成一个灵敏度分析报告 |
| Options | 设置系统选项 |

续表

| 命令 | 功能 |
|---|---|
| Generate | 生成当前模型的代数表达式 |
| Picture | 显示模型矩阵图 |
| Debug | 跟踪不可行或无界线性规划中的表达式错误 |
| Model Statistics | 显示关于模型中主要技术细节的报告 |
| Look | 为当前窗口产生一个含有模型表达式的报告 |

表 5-4　Window 菜单的主要命令

| 命令 | 功能 |
|---|---|
| Command Window | 打开一个 LINGO 命令行命令窗口 |
| Status Window | 打开一个求解状态窗口 |
| Send to Back | 将当前窗口放到所有窗口的后面 |
| Close All | 关闭所有打开的窗口 |
| Tile | 将所有打开的窗口排列成平铺的形式 |
| Cascade | 将所有打开的窗口排列成层叠的形式 |
| Arrange Icons | 对齐主窗口底部的所有成为图标的窗口 |

表 5-5　Help 菜单的主要命令

| 命令 | 功能 |
|---|---|
| Help Topics | 检索 LINGO 的帮助信息 |
| Register | 在线注册 LINGO |
| Auto Update | 在有新版本时提示下载升级的软件 |
| About LINGO | 显示 LINGO 的版本及其对应模型的最大规模以及如何联系 LINDO System 公司的信息 |

## 5.3　窗口命令详细介绍

### 5.3.1　File 菜单

LINGO 的 File 菜单如图 5-3 所示，下面对 File 菜单的各项命令进行详细介绍。

#### 5.3.1.1　File|New

New 命令（快捷键 F2）可以打开一个新的空白窗口。当用户选择 New 时，会看到图 5-4 所示对话框。

该对话框中列出了可以选择创建的四种文件类型。

1. LINGO Model（*.lg4）

LG4 格式是在 LINGO 4.0 版本中建立的，是 LINGO 在 Windows 操作系统下存储模型的首


图 5-3 File 菜单

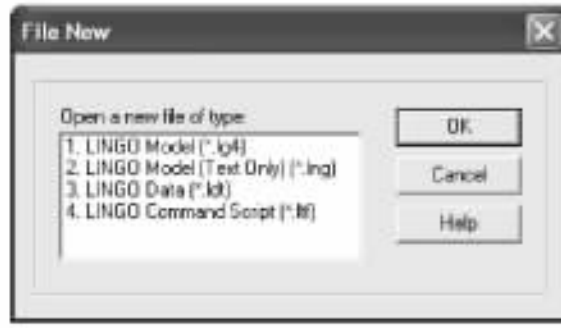



图 5-4 New 对话框

选格式。这种格式支持多种字体、自定义格式和对象的链接与嵌入（OLE）。LG4 文件以专用的二进制格式存放在磁盘中，因此，这些文件不能被其他应用程序直接读取，也不能传输到除 PC 以外的其他平台上。

2. LINGO Model (Text Only)（*.lng)

LNG 格式是一种较灵活的存储模型的格式。它是 LINGO 4.0 以前的版本应用的标准文件格式，并且还应用于除 Windows 以外的其他平台。LNG 文件以 ASCII 文本保存在磁盘里，因此可以被任何支持文本文件的应用程序和文本处理器读取。LNG 文件也可以传输到除 PC 以外的平台上，但 LNG 文件不支持多种字体、自定义格式和 OLE。

3. LINGO Data（*.ldt)

LDT 文件是数据文件，一般用 @FILE 函数将其调入到 LINGO 模型中。@FILE 函数仅仅可以读取文本文件，考虑到这一点，所有的 LDT 文件以 ASCII 文件形式存储。同样，LDT 文件不支持多种字体、自定义格式和 OLE。

4. LINGO Command Script（*.ltf)

LTF 文件由 LINGO 命令脚本组成。它是包含一系列 LINGO 命令的 ASCII 文本文件，而且这些 LINGO 命令可以通过 File | Take Commands 命令执行。LTF 文件不支持多种字体、自定义格式和 OLE。

当直接单击 New 工具栏按钮或 F2 键时，LINGO 将默认为新建模型文件，因此，LINGO 并不显示文件类型对话框，而是立即打开一个 LG4 类型的模型文件。

如果已经用 LINGO | Options 命令将默认文件类型 LG4 改成 LNG，那么当用户单击 New 命令或按 F2 键时，LINGO 将自动打开一个 LNG 类型的模型文件。

5.3.1.2 File|Open...

Open 命令(快捷键 F3)可以从磁盘中读取已保存的文件并将其置入 LINGO 窗口中。这个文件可以是 LINGO Model (*.lg4)文件,也可以是其他任何 ASCII 文本格式的文件。执行 Open 命令后,将会看到图 5-5 所示对话框。


图 5-5 Open 对话框

在文件名(File name)编辑框中输入文件名或者从现有的文件列表中通过选择一个文件名,然后单击 Open 按钮将打开这个文件;单击 Cancel 按钮不打开文件并退出;单击 Help 按钮可寻求帮助。

用户还可以从文件类型(Files of type)列表框中选择另外一种文件类型,这样 LINGO 将只列出这种类型的文件。

如果打开的文件有.MPS 扩展名,那么 LINGO 将使用 MPS 阅读器来解析文件。MPS 文件格式是 IBM 公司开发的一种企业标准格式,用来从一个求解程序或平台向另一个求解程序或平台传输模型。当导入一个 MPS 文件时,LINGO 就会从磁盘中读取这个 MPS 文件并将其转换成相应的 LINGO 模型放至一个新的模型窗口中。

5.3.1.3 File|Save

Save 命令(快捷键 F4)用于将活动窗口中的内容以窗口上现有的文件名保存到磁盘中。如果这个窗口未曾保存过,系统将提示输入一个文件名后再进行保存。

5.3.1.4 File|Save As...

Save As 命令(快捷键 F5)要求以一个新的文件名保存活动窗口中的内容。当选择 Save As 命令时,将看到图 5-6 所示对话框。

在保存之前可以在 File name 编辑框中输入一个新的文件名,或从现存的文件列表中通过双击选择一个文件名。如果没有指定一个文件扩展名,LINGO 将自动追加默认模型格式的扩展名。如果想阻止 LINGO 追加默认扩展名,就把文件名放在双引号中。


图 5-6　Save As 对话框

如前所述，点击 Save 按钮保存模型，点击 Cancel 按钮不保存并退出，点击 Help 按钮提供帮助。

用户还可以从 Save as type 列表框中选择不同的文件类型。如果模型中指定了字体或嵌入了对象，就必须用 LG4 文件格式保存它们。如果想创建一个模型的副本，那么就用 LNG 文件格式。

5.3.1.5　File | Close

Close 命令（快捷键 F6）用于关闭活动（最前面的）窗口。如果窗口已经被修改但没有保存，系统将会提示是否保存更改。

5.3.1.6　File | Print...

Print 命令（快捷键 F7）用于将活动窗口中的内容发送到打印机。LINGO 将显示图 5-7 所示的 Print 对话框。

从 Name 列表框中选择使用的打印机，单击 Properties 按钮可以更改打印机的属性，在 Print range 组框中选择打印页数范围。如果想打印多个副本，那么在 Number of copies 栏输入需要的副本数并指定是否需要副本校对（假如打印机有校对功能）。最后单击 OK 按钮开始打印，或单击 Cancel 按钮不打印并退出。

5.3.1.7　File | Print Setup...

Print Setup 命令（快捷键 F8）用于配置打印机。LINGO 将显示图 5-8 所示 Print Setup 对话框。

从 Paper 组框中选择纸张类型和来源，在 Orientation 组框中选择输出的方式。单击 Cancel 按钮退出且不改变打印机配置，单击 OK 按钮保存更改并退出 Print Setup 窗口。

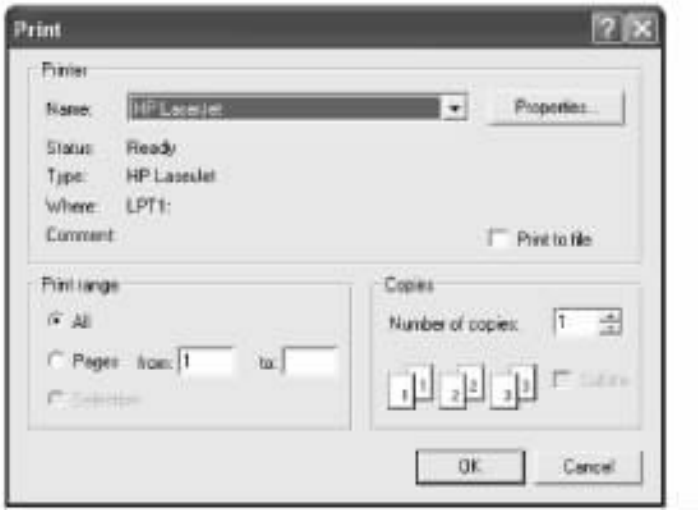



**图** 5-7　Print **对话框**

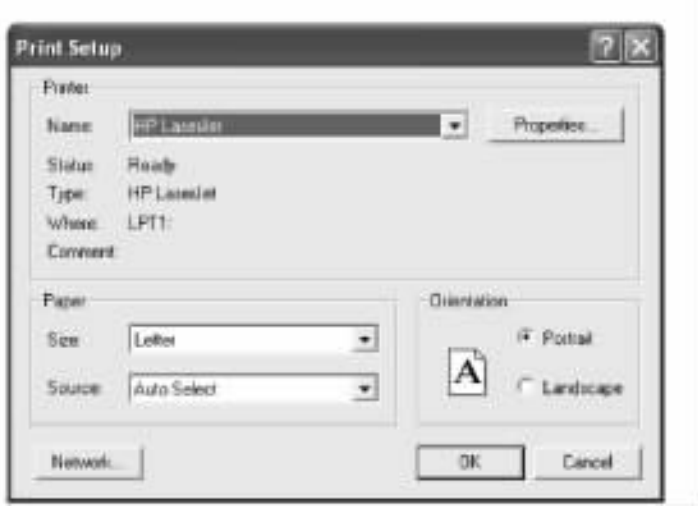



**图** 5-8　Print Setup **对话框**

5.3.1.8　File|Print Preview

Print Preview**命令 (快捷键** Shift + F8**) 可以将活动窗口中的内容按打印时的样式显示。运行** Print Preview**命令后,活动窗口中的内容将显示在预览窗口中,见图** 5-9 **所示窗口。**

**窗口中各按钮的功能如下：**

Print**——将文件输送到打印机；**

Next Page**——浏览下一页；**

Prev Page**——浏览前一页；**

One Page**——使浏览器进入单页显示模式,此时该按钮显示为** Two Page**；**

Two Page**——使浏览器进入双页显示模式,此时该按钮显示为** One Page**；**

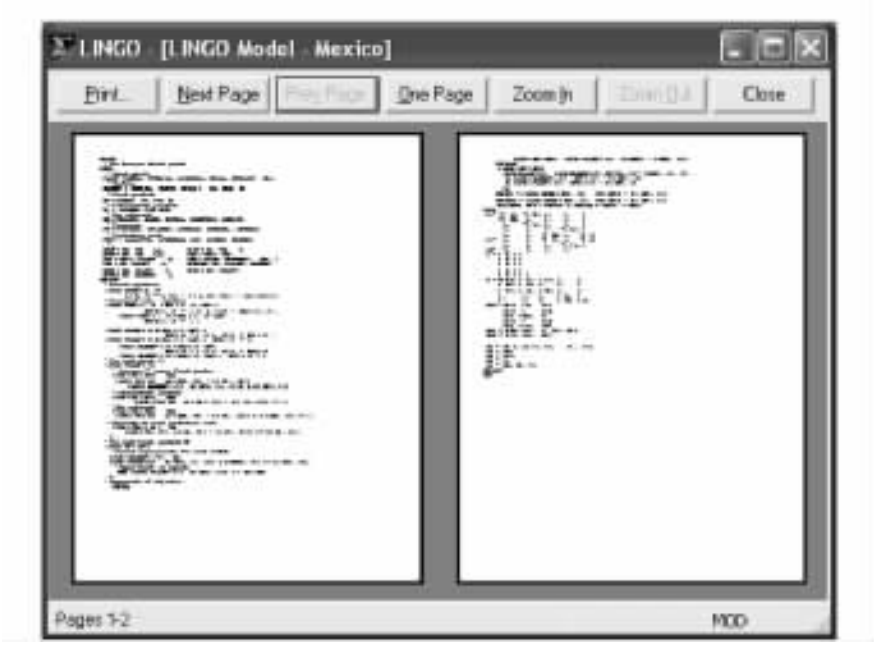



图 5-9　Print Preview 窗口

Zoom In——使浏览器聚焦于文本中的某个区域；

Zoom Out——撤销 Zoom In 命令；

Close——关闭打印预览并返回到正常的 LINGO 命令模式。

5.3.1.9　File|Log Output...

一般情况下，LINGO 是在菜单驱动模式下操作，即选择下拉菜单中的命令并在单独窗口中显示报告。LINGO 也可以在命令模式下操作，即以文本命令或脚本命令驱动应用程序，并且将所有的输出传到命令窗口。这样输出量将会十分有限。如果需要将命令窗口中的所有显示内容做磁盘备份，就可以使用 Log Output 命令（快捷键 F9）。Log Output 命令将打开一个对话框，在这个对话框中可以为日志文件命名，而且可以通过选择 Echo to screen 复选框实现同时将输出内容送回命令窗口。如果希望将输出内容加到日志文件的结尾处，可以通过选择 Append output 复选框实现。

当选择一个日志文件时，File 菜单中的 Log Output 命令前面就会出现复选标记，再单击一次这个命令就可以关闭 Log Output，复选标记消失。

5.3.1.10　File|Take Commands...

Take Commands 命令（快捷键 F11）用于处理 LINGO 命令脚本文件。这里以建立一个小的脚本文件为例介绍 Take Commands 命令的用法。

首先，运行 New 命令。在 New 对话框中选择 LINGO Command Script，单击 OK 按钮，LINGO 将打开一个空的脚本文件。

然后将图 5-10 所示内容输入到脚本文件中。该命令脚本文件的功能是输入一个产品混

合模型，求解并将报告送入一个文本文件。用 File | Save 命令将这个命令脚本文件以文件名 MyScript.ltf 保存到磁盘中。

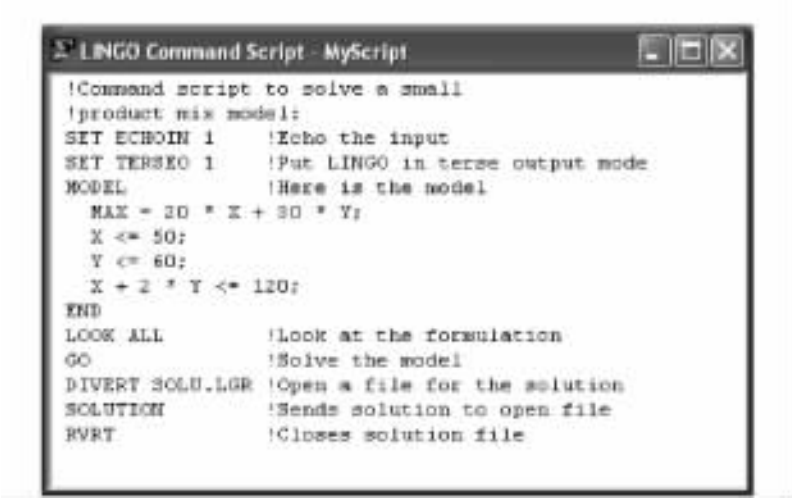



图 5-10　命令脚本文件

这时，选择 Take Commands 命令，出现图 5-11 所示的对话框。

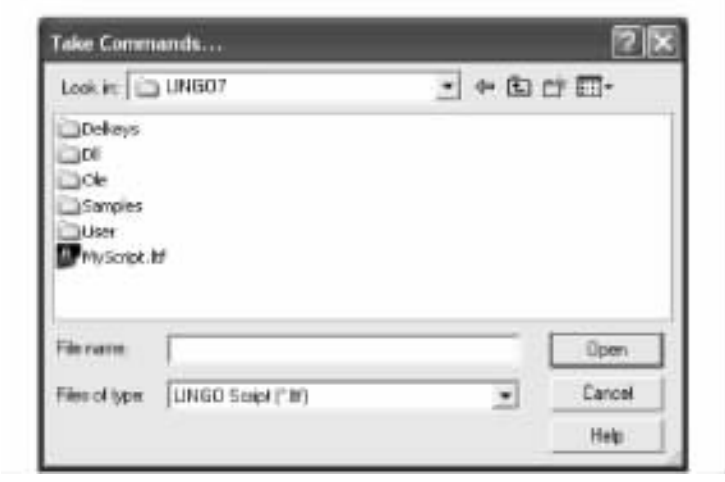



图 5-11　Take Commands 对话框

双击 MyScript.ltf 图标，就会出现 LINGO 的命令窗口，并且通过观察该命令窗口中出现的命令和输出可以看到 LINGO 处理这个脚本文件的过程。当 LINGO 处理完命令脚本中的所有命令后，就会看到图 5-12 所示的窗口。

值得注意的是，结果报告 SOLU.lgr 也是由该命令脚本创建的。如果打开这个文件，就会发现下面的模型结果报告：

```
      Variable           Value        Reduced Cost
             X        50.00000            0.000000
             Y        35.00000            0.000000
```


**图 5-12　命令脚本处理结果**

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 2050.000 | 1.000000 |
| 2 | 0.000000 | 5.000000 |
| 3 | 25.00000 | 0.000000 |
| 4 | 0.0000000 | 15.00000 |

5.3.1.11　File | Import LINDO File...

Import LINDO File 命令（快捷键 F12）的功能是将以 LINDO 的 LTX 格式（即文体格式）保存的 LINDO 文件导入到模型文件中。当一个 LINGO 文件导入了一个 LINDO 文件时，LINGO 会自动对模型文本作更改，使其符合 LINGO 的语法要求，而这些语法要求与 LINDO 是不同的。

需要注意的是，Import LINDO File 命令一般用于导入一个小型模型。要想导入一个大型 LINDO 模型，该模型必须用 MPS 格式保存，并且通过 Open 命令将这个模型导入到 LINGO 中。

例如，假设一个名为 MyModel.ltx 的小型 LINDO 模型如下：

```
！一个小型人员安排模型：
MIN    100 XMON + 100 XTUE + 100 XWED +
  100 XTHU + 100 XFRI + 100 XSAT + 100 XSUN
SUBJECT TO
SUN)  XWED + XTHU + XFRI + XSAT + XSUN > = 18
```

```
MON)  XMON + XTHU + XFRI + XSAT + XSUN > = 16
TUE)  XMON + XTUE + XFRI + XSAT + XSUN > = 15
WED)  XMON + XTUE + XWED + XSAT + XSUN > = 16
THU)  XMON + XTUE + XWED + XTHU + XSUN > = 19
FRI)  XMON + XTUE + XWED + XTHU + XFRI > = 14
SAT)  XTUE + XWED + XTHU + XFRI + XSAT > = 12
END
```

**如果运行** Import LINDO File **命令导入** MyModel.ltx,LINGO **将装载这个** LINDO **文件并作一些修改,所以将看到如下内容:**

```
MIN = 100 * XMON + 100 * XTUE + 100
 * XWED + 100 * XTHU + 100 * XFRI +
100 * XSAT + 100 * XSUN;
   [SUN] XWED + XTHU + XFRI + XSAT + XSUN > 18;
   [MON] XMON + XTHU + XFRI + XSAT + XSUN > 16;
   [TUE] XMON + XTUE + XFRI + XSAT + XSUN > 15;
   [WED] XMON + XTUE + XWED + XSAT + XSUN > 16;
   [THU] XMON + XTUE + XWED + XTHU + XSUN > 19;
   [FRI] XMON + XTUE + XWED + XTHU + XFRI > 14;
   [SAT] XTUE + XWED + XTHU + XFRI + XSAT > 12;
```

**可见,为了满足语法要求,**LINGO **作了如下改动:**

1) MIN **后增加了一个等号(=);**

2) **在模型中有乘法运算的地方增加了乘法符号(*);**

3) **从模型中删除了** SUBJECT TO **一行;**

4) **每个表达式后都增加了分号;**

5) **约束名被加上了方括号。**

LINGO **对** LINDO **模型所作的修改在模型中并不作说明。**

5.3.1.12 File|Export File

Export File **命令要求输入** MPS **或** MPI **扩展名,并将文件以该扩展名的类型输出。**

5.3.1.13 File|Database User Info

**单击** File **菜单下的** Database User Info **命令,弹出图** 5-13 **所示对话框。**LINGO **允许用** @ODBC() **函数直接与数据库连接。通常情况下,连接到模型数据库需要一个用户** ID **和密码。为了避免每一次运行模型都必须输入用户** ID **和密码,可以在开始第一次会话时运行此命令并只需输入一次。**

**为了安全起见,从这次会话到下一次会话** LINGO **并不保存这些信息。因此,在每一次会话开始时都必须运行这个命令。如果不考虑安全,也可以保存这些信息,创建一个含有** DBUID **命令和** DBPWD **命令的** AUTOLG.DAT **文件。**AUTOLG.DAT **文件中的命令会在每一次** LINGO **开**


图 5-13　Database User Info 对话框

始时自动运行。这样，包含在 AUTOLG. DAT 文件中 DBUID 和 DBPWD 命令会在每一次 LINGO 开始时恢复数据库用户信息。

## 5.3.2　Edit 菜单

LINGO 的 Edit 菜单如图 5-14 所示。这个菜单包含关于编辑和修改窗口中文本的一般命令。


图 5-14　Edit 菜单

### 5.3.2.1　Edit | Undo

Undo 命令（快捷键 Ctrl + Z）用于撤销对窗口中的内容所作的最后一次修改，但是 Undo 不能撤销拖放文本移动的操作。LINGO 只能储存有限数量的撤销操作，所以不能依赖 LINGO 撤销大量的更改。

### 5.3.2.2　Edit | Redo

Redo 命令（快捷键 Ctrl + Y）将恢复最后一次撤销的操作。LINGO 只能储存有限数量的恢复操作，所以也不能依赖 LINGO 恢复大量的更改。

### 5.3.2.3　Edit | Cut

Cut 命令（快捷键 Ctrl + X）用于清除选中的文本块并将其放到剪贴板上用于粘贴。选择要剪切的文本块，可以将光标放在文本块的开头处按下鼠标左键，拖动鼠标直至光标到该文本块的最后，此时文本块以反色显示，这样就可以用 Cut 命令将文本块移出文档，放至 Windows 剪贴板上。

### 5.3.2.4　Edit | Copy

Copy 命令（快捷键 Ctrl + C）用于将选中的文本块复制到剪贴板上用于粘贴。选择文本块的方法参考 Cut 命令。

5.3.2.5 Edit|Paste

Paste 命令(快捷键 Ctrl + V)用于将 Windows 剪贴板上的内容代替活动窗口中选择的内容。Paste 命令和 Copy 命令相互配合,是从其他应用程序向 LINGO 模型中输入少量数据的便利方法。

5.3.2.6 Edit|Paste Special...

Paste Special 命令用于将 Windows 剪贴板上的内容插入到活动窗口的光标处。这个命令不仅可以和 Paste 一样插入简单的文本,还可以用于插入其他对象和连接其他对象,这一点在为模型增加与其数据的连接时特别有用。通过插入与一个数据源的连接,可以很容易地找到并浏览它们。

例如,有如下运输模型:

```
!一个 3 个 WAREHOUSE 和 4 个 CUSTOMER 的模型问题;
SETS:
  WAREHOUSE/WH1,WH2,WH3/:CAPACITY;
  CUSTOMER/C1,C2,C3,C4/:DEMAND;
  ROUTES(WAREHOUSE,CUSTOMER):COST,VOLUME;
ENDSETS
!目标函数;
MIN = @SUM(ROUTES:COST * VOLUME);
!需求约束;
@FOR(CUSTOMER(J):
  @SUM(WAREHOUSE(I):VOLUME(I,J)) > =
    DEMAND(J));
!供给约束;
@FOR(WAREHOUSE(I):[SUP]
  @SUM(CUSTOMER(J):VOLUME(I,J)) < =
  CAPACITY(I)));
!参数;
DATA:
  CAPACITY = @OLE('D:\LNG\TRANLINKS.XLS');
  DEMAND = @OLE('D:\LNG\TRANLINKS.XLS');
  COST = @OLE('D:\LNG\TRANLINKS.XLS');
  @OLE('D:\LNG\TRANLINKS.XLS') = VOLUME;
ENDDATA
```

从数据域里可以看到,该模型通过 Excel 文件 TRANLINKS.XLS 输入数据,并将结果送回该文件。因此,可以在模型中插入一个电子表格,这样不用启动 Excel 就可以浏览数据和结果。为了做到这一点,首先打开 Excel 并装载电子表格,过程如图 5-15 所示。

在电子表格里选择区域 B2:F21,然后从 Excel 中复制该区域,接下来回到 LINGO 窗口,将

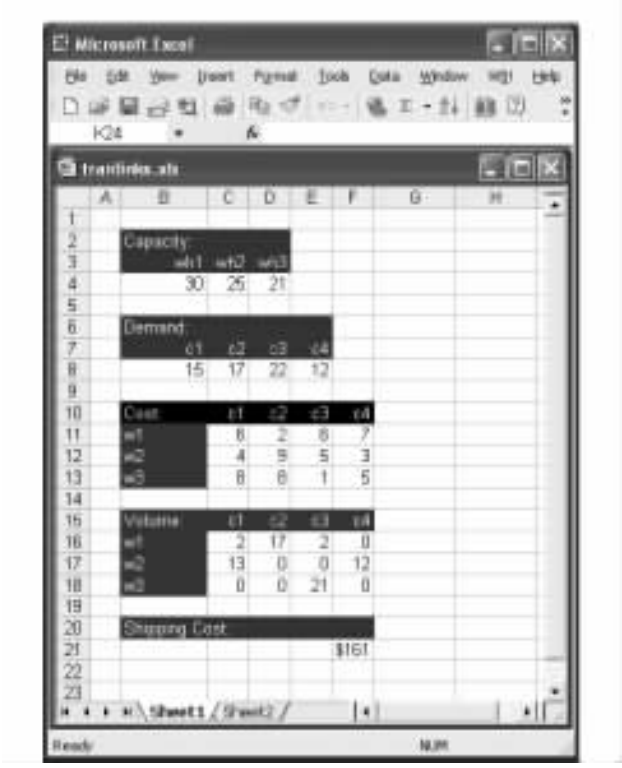



**图** 5-15　Excel **模型数据**

**光标放在数据域前，选择** Edit | Paste Special **命令并在对话框中单击** Paste Link **选项，如图** 5-16 **所示。**

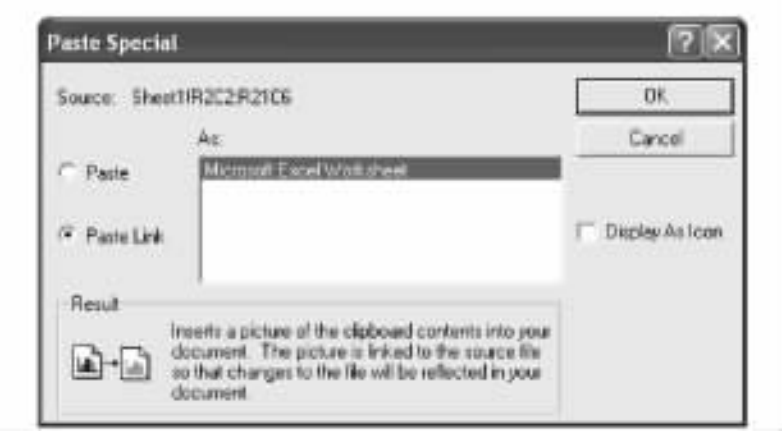



**图** 5-16　Paste Special **对话框**

**最后，单击** OK **按钮就会在** LINGO **模型中看到图** 5-17 **所示电子表格的内容。**

**这个连接将作为** LINGO **文件的一部分被保存起来。因此，无论什么时候打开该模型，都可以看到此电子表格。**

**值得注意的是，当重新打开模型时，可能也想打开连接，这样内容就能够自动更新了。要**


**图 5-17　LINGO 中的 Excel 表格**

**做到这一点，可以选择 LINGO 模型里的电子表格，然后执行 Edit | Links 命令，单击对话框中的 Open Links 按钮就可以了。**

**最后要注意的是，所有嵌入的连接和对象都将被 LINGO 编辑器忽略。因此，模型的任何位置都可以自由插入连接和对象。**

5.3.2.7　Edit | Select All

Select All **命令用于选择活动窗口中的全部内容。当准备把窗口中的全部内容复制到其他地方或删除此窗口中的全部内容时，将会用到这个命令。**

5.3.2.8　Edit | Find...

Find **命令（快捷键** Ctrl + F**）用于在活动窗口中的文本中查找需要的字符串。当运行** Find **命令时，将看到图** 5-18 **所示对话框。**

**在** Find what **文本框中输入想查找的文本，选择** Match whole word only **复选框，**LINGO **仅查找与整个词匹配的文本；选择** Match case **复选框，**LINGO **仅查找具有相同大小写字母的文本；单击** Find Next **按钮开始查找下一处匹配位置。**

5.3.2.9　Edit | Find Next

Find Next **命令（快捷键** Ctrl + N**）用于查找下一处匹配文本，该文本是在最近一次使用** Find


图 5-18　Find 对话框

命令时刚查找过的。

5.3.2.10　Edit|Replace...

Replace 命令(快捷键 Ctrl + H)用于将活动窗口中的字符串用另一字符串替换。运行 Replace 命令后,将看到图 5-19 所示对话框。

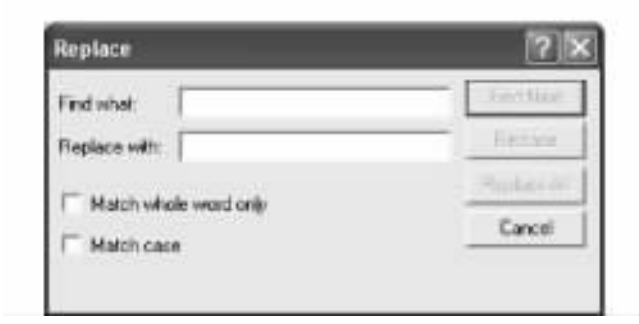



图 5-19　Replace 对话框

在 Find what 文本框中输入准备替换掉的文本,在 Replace with 文本框中输入准备替换旧文本的新文本,点击 Find Next 按钮,LINGO 将找到下一处匹配的旧文本;单击 Replace 按钮,下一处旧文本的内容将被新文本代替;单击 Replace All 按钮,文档中全部匹配的旧文本将被新文本代替。

5.3.2.11　Edit|Go To Line...

Go To Line 命令(快捷键 Ctrl + T)用于跳转到活动窗口的指定行数。当运行 Go To Line 命令时,将看到图 5-20 所示对话框。

首先,在 Go to line number 编辑框中输入一个行号,然后点击 OK 按钮,LINGO 将跳转至该行;点击 Top 按钮,将跳转至文档顶部;点击 Bottom 按钮,将跳转至文档底部。

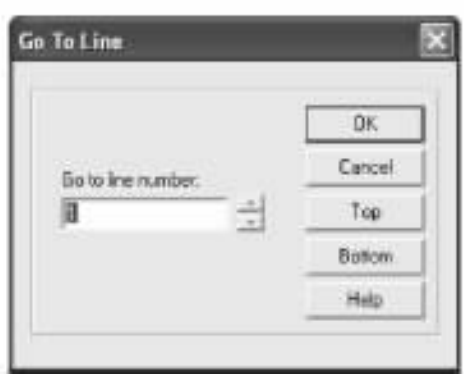



图 5-20　Go To Line 对话框

5.3.2.12　Edit|Match Parenthesis

在文档中选择一个括号,然后用 Match Paren-

thesis 命令(快捷键 Ctrl + P)可以查找与所选括号匹配的另一半括号。假设使用了如下嵌套语句:

```
@ FOR (FXA (I,J) :
  JP (I,J) = MPF (I) * CAGF (I,J) ;
  JP (I,J) = MPA (J) * CFGA (I,J)) ;
```

在这样的语句中查找一个括号的匹配括号非常困难,这时 Match Parenthesis 命令就派上了用场。如果在运行 Match Parenthesis 命令之前没有选定一个括号,那么 LINGO 将选择与光标所在位置最近的括号。

除这个命令之外,还有另一个查找匹配括号的方法。当 LINGO | Options 命令下的 Paren Match 选项被启用时,LINGO 将用红色高亮显示匹配的括号。具体操作时,将光标紧跟着放在某个括号之后,LINGO 将用红色同时显示与其匹配的括号。这些括号一直用红色显示,直到将光标移至其他地方为止,此时它们将返回黑色状态。

5.3.2.13 Edit | Paste Function

Paste Function 命令用于在当前插入点处粘贴任意的 LINGO 内部函数,从二级菜单中可以选择想要粘贴的 LINGO 函数种类。

如图 5-21 所示,已经从二级菜单中选择了 External Files 类,在菜单的最右面是所有用于处理外部文件的函数。选择其中一个函数,LINGO 将在模型中粘贴一个该函数的模板,并为每个函数自变量提示占位符,然后就可以用模型中相应的自变量代替占位符。


图 5-21 Paste Function 下拉菜单

5.3.2.14 Edit | Select Font...

Select Font 命令用于为选中的文本选择新的字体、大小、样式、颜色或显示效果。如果选择的是等宽字体格式(如 Courier),将会使阅读模型和结果报告变得很容易。只有当文件以 LG4 格式保存时自定义字体才会被保存。

值得注意的是,如果语法的颜色格式是打开的,那么就不能改变文本的显示颜色。如果想在文档中指定显示颜色,那么就要关闭语法的颜色格式。

5.3.2.15　Edit|Insert New Object...

Insert New Object 命令用于在一个模型里插入一个对象或将一个对象连接到模型里，如同 Edit|Paste Special 命令，有助于模型与数据源的连接。但是，Paste Special 命令仅仅可以与外部对象的一部分连接，而 Insert New Object 命令可以与整个对象连接。现举例说明。假设有图 5-22 所示员工安排模型。在模型的数据域里，用 @ODBC 函数从 ODBC 数据源 STAFFING 里为属性 NEED 输入数据，并用 ODBC 函数将属性 START 的最优值送回到同样的数据源中。因为这个数据源是模型中一个完整的部分，所以最好在模型里放置一个与它的连接，这样每次需要它时，都可以很容易重新找到。这点可以用 Edit|Insert New Object 命令实现，操作步骤如下：


图 5-22　员工安排模型

1）将光标放在准备放置连接图标的位置（注意：LINGO 编辑器忽略与外界对象的任何连接，所以可以在模型的任何位置插入连接）；

2）运行 Edit|Insert New Object 命令，将看到如图 5-23 所示对话框；

3）选择 Create from File 单选框，将看到如图 5-24 所示对话框；

4）在文本框中输入或点 Browse 按钮，选择包含数据的数据库名；

5）选择 Display As Icon 复选框，此时对话框如图 5-24 所示；

6）最后，单击 OK 按钮，一个表示连接数据库的图标就会出现在 LINGO 的模型窗口里，如图 5-25 所示。

这样无论什么时候想编辑和浏览该数据源，双击图标即可。在这个例子中，Microsoft Access 将启动并装载该数据库，如图 5-26 所示。

最后应注意，只有当模型以 LG4 格式保存时，连接的对象才会被保存。

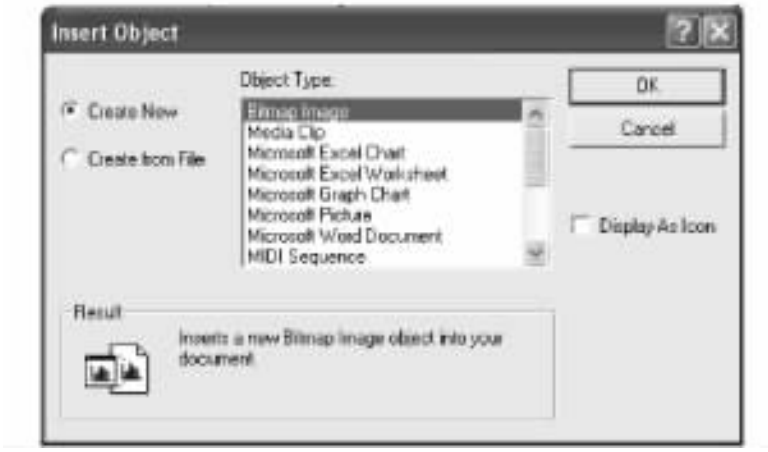



**图** 5-23　Insert Object **对话框**

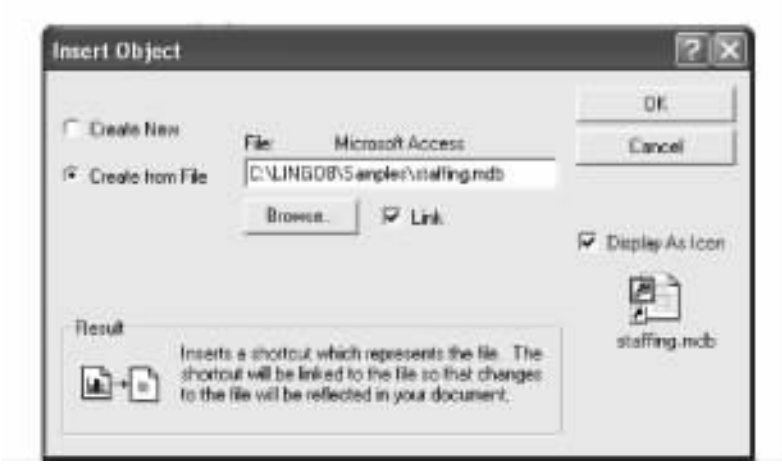



**图** 5-24　Insert Object **对话框**

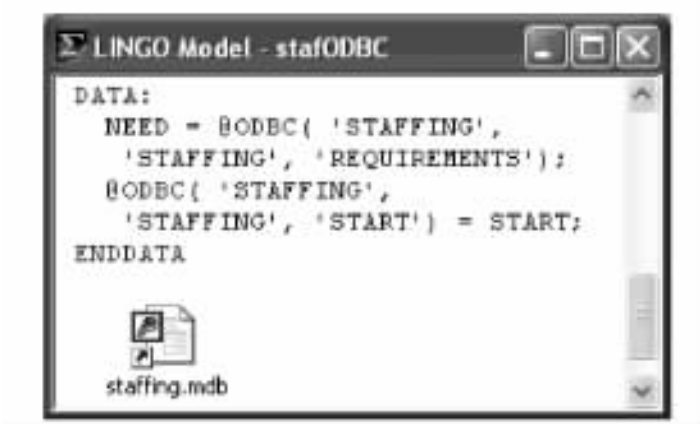



**图** 5-25　**修改的员工安排模型**

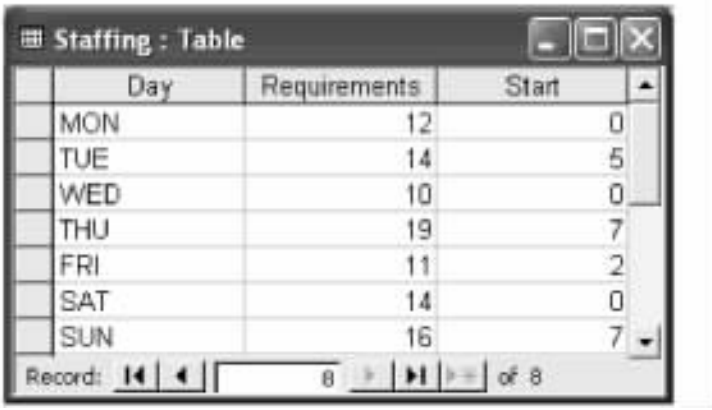



**图** 5-26　Access **数据表**

5.3.2.16　Edit|Links...

Links **命令用于在** LINGO **文档里修改与外部对象连接的属性,对话框如图** 5-27 **所示。**

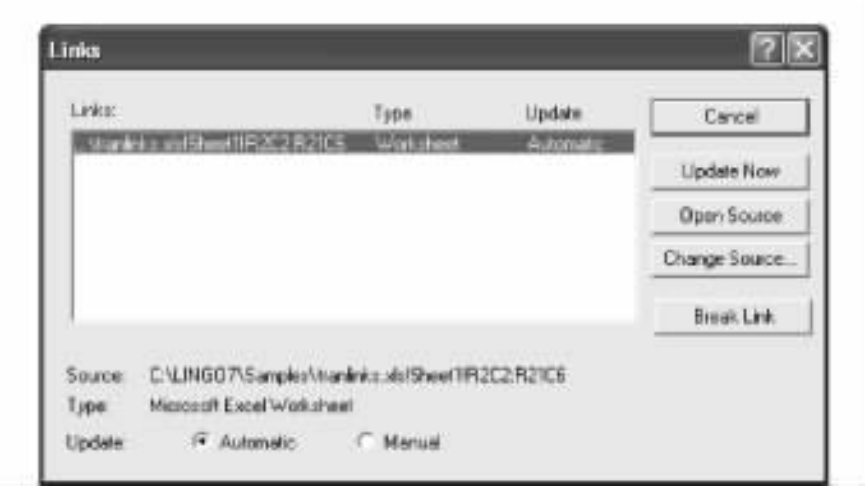



**图** 5-27　Links **对话框**

**如果选择** Automatic **单选按钮,当源文件被改变时,**LINGO **能自动更新对象;而选择** Manual **单选按钮后,只有点击** Update Now **按钮才能更新对象。**Open Source **按钮用于打开一个自动连接。一旦连接被打开,数据源中的任何变化都将反映在** LINGO **模型里。**Change Source **按钮用于改变所对应的数据源。**Break Link **按钮用于断开与外部数据源间的连接。**

5.3.2.17　Edit|Object Properties

**在模型中选择一个已连接的或嵌入的对象,然后就可以用** Object Properties **命令(快捷键** Alt + Enter**)来修改这个对象的属性,可以改变的属性包括对象的显示、对象的源、更新方式(自动或手动)、打开对象的连接、更新对象以及断开对象连接。**

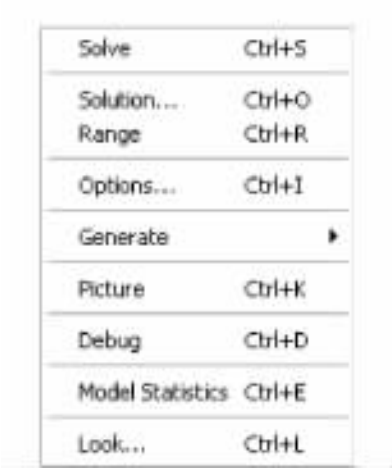



图 5-28 LINGO 菜单

### 5.3.3 LINGO 菜单

LINGO 菜单如图 5-28 所示，这个菜单包含求解模型和生成结果报告的命令以及定制 LINGO 配置的 Option 命令。

#### 5.3.3.1 LINGO | Solve

Solve 命令（快捷键 Ctrl + S）用于求解活动窗口中的模型，并且仅仅用在模型窗口，结果报告、脚本和数据窗口不能使用该命令。

当求解一个模型时，LINGO 首先检查模型的语法。如果发现语法错误，就会看到图 1-4 所示的对话框。在 Error Text 框中，LINGO 显示出现错误的行号以及这一行的文本，并指出错误发生的位置。在大多数情形下，LINGO 能很好地指出错误发生的位置。然而有时错误并不在 LINGO 所指的位置上，因此还必须检查其相邻行以寻找可能出现的错误。

当运行 Solve 命令后（假定模型没有出现语法错误），LINGO 将显示如图 1-5 所示的求解状态窗口。这个窗口包含模型的各个组成部分以及求解的过程。模型求解完成后，LINGO 会创建一个包含结果报告的新窗口，可以通过拖动滚动条来检查它的内容，也可以将其保存为一个文本文件或打印。

#### 5.3.3.2 LINGO | Solution...

Solution 命令（快捷键 Ctrl + O）用于为当前窗口产生一个结果报告。这个结果报告可能是文本形式也可能是图表形式。选择模型窗口，运行 LINGO | Solution 命令，将看到图 5-29 所示对话框。

在 Attribute or Row Name 列表框中选择一个属性名或行名。如果没有在列表中选择一个名字，LINGO 将生成一个包括所有属性和行的结果报告。

在 Header Text 文本框中输入标题性文本（例如 values for X），则这些文本将出现在报告顶部。

在 Type of Output 单选框中，可以选择文本（Text）或图表（Graph）形式输出。如果选择文本（Text），LINGO 将生成一个包含文本格式的求解结果的新窗口；如果选择图表（Graph），LINGO 将生成一个用某种图表格式表示的求解结果的新窗口。当前提供的图表格式有 Bar（柱状图）、Line（折线图）和 Pie（饼图）。

选择 Nonzeros Only 复选框，将看到仅包含非零变量和有效约束的报告。

如果已经选择了图表形式显示，Graph Properties 标题框将不再是灰色，这时就可以选择图的显示样式。在 Values 框中，可以选择以图表显示 Primal（原始）或 Dual（对偶）值。Bounds 框中提供了设置图表中显示值范围的选项。如果在 Lower 框中输入一个数，LINGO 将在图中仅仅显示大于等于这个值的点。如果在 Upper 框中输入一个数，LINGO 将在图中仅仅显示小于等于这个值的点。如果选择了 Include Labels 复选框，LINGO 将把图表的每一点都贴上相应的

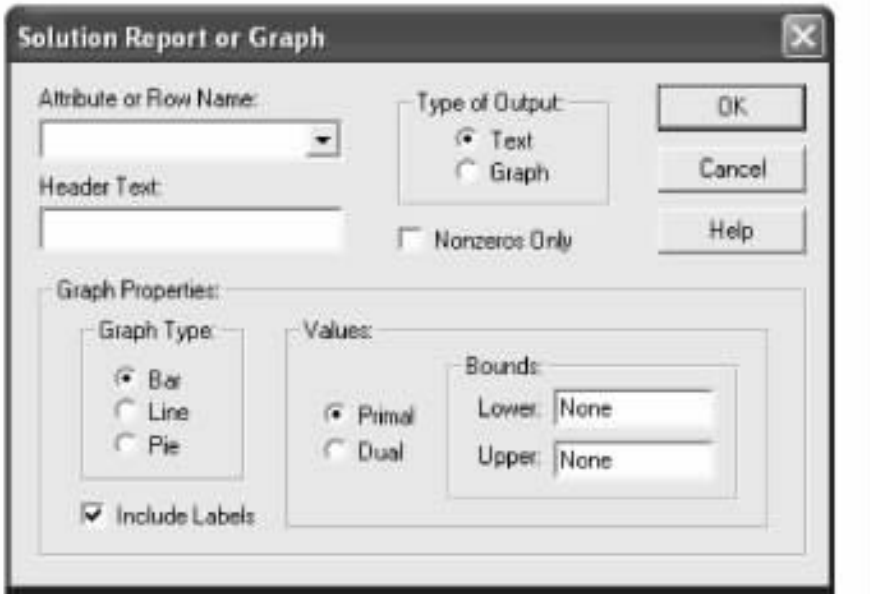



图 5-29　Solution 对话框

变量名。

选择完毕后，单击 OK 按钮，LINGO 就会创建一个结果报告窗口。

值得注意的是，LINGO 仅仅将结果报告存放在内存里，并且只存放最近一次 LINGO | Solve 命令生成的结果报告。如果试图运行 Solution 命令而 LINGO 尚没有相应的求解结果，就会收到一个错误信息。因此，如果打算使用两个或更多的模型且求解时间很长，最好将求解报告都保存在磁盘中，这样以后不需要重新求解模型就可以查阅它们了。

#### 5.3.3.3　LINGO | Range

Range 命令（快捷键 Ctrl + R）用于为当前活动窗口生成一个灵敏度报告。一个灵敏度分析报告会显示：

1）在决策变量最优值不变的情况下，目标函数系数的变化范围；

2）在对偶价格和下降成本都不变的前提下，约束条件右边的常数的变化范围。

注意：当求解模型时，必须启用灵敏度计算（Range computations）功能，求解程序才会计算灵敏度值。灵敏度计算在默认状态下是关闭的，要想启用它，就必须运行 LINGO | Options 命令，选择 General Solve 页，并在 Dual Computations 列表框中选择 Prices and Ranges 选项。但是，灵敏度计算占用很多额外的计算时间。

例如，为模型

```
[OBJECTIVE] MAX = 20 * A + 30 * C;
[ALIM]      A < = 60;
[CLIM]      C < = 50;
[JOINT]     A + 2 * C < = 120;
```

产生的灵敏度报告如下：

Ranges in which the basis is unchanged:

Objective Coefficient Ranges

| Variable | Current Coefficient | Allowable Increase | Allowable Decrease |
|---|---|---|---|
| A | 20.00000 | INFINITY | 5.000000 |
| C | 30.00000 | 10.00000 | 30.00000 |

Righthand Side Ranges

| Row | Current RHS | Allowable Increase | Allowable Decrease |
|---|---|---|---|
| ALIM | 60.00000 | 60.00000 | 40.00000 |
| CLIM | 50.00000 | INFINITY | 20.00000 |
| JOINT | 120.0000 | 40.00000 | 60.00000 |

灵敏度报告第一部分的标题是目标函数系数变化范围(Objective Coefficient Ranges)。第一列标题为 Variable,列出了所有可最优化的变量,即决策变量;第二列标题为 Current Coefficient,列出了决策变量在目标函数里的系数;第三列标题为 Allowable Increase,列出了在不改变决策变量最优值的前提下,其系数可上调的界限;最后一列标题为 Allowable Decrease,列出了在决策变量最优解不变的前提下,其系数可下调的界限。当需要回答诸如"如果增加(或减少)这项活动,可以增加(或减少)多少收益"之类的问题时,有关目标函数系数变化的上下界数据是很有帮助的。

从以上例子目标函数系数的灵敏度报告中可以看出:只要变量 A 的系数大于等于 15,最优值就保持不变。同样,变量 C 的系数在[0,40]范围内变化时,最优值也不会改变。

注意:仅仅在改变一个目标函数的系数或约束条件右面的常数时,灵敏度分析才有效。当两个或两个以上的系数同时发生变化时,LINGO 提供的灵敏度分析报告是无效的。此外,灵敏度变化范围仅仅是最优解保持不变的一个较低的限度。在这个范围内改变系数的值,最优值保持不变。如果超出了灵敏度范围,最优值是否变化还不能确定。

灵敏度分析报告第二部分的标题是右面常数变化范围(Righthand Side Ranges)。第一列标题为 Row,列出了模型中所有可优化的约束名。第二列标题为 Current RHS,列出了每一个约束条件右边常数项的值。接下来的两列标题分别为 Allowable Increase 和 Allowable Decrease,列出了在对偶价格和下降成本的最优值保持不变的前提下,右边常数项允许的变化范围。对偶价格就是影子价格,表示应该用什么价格卖出或买入资源。然而,对偶价格不能表示用这个价格卖出或买入多少资源,这些信息可以从约束条件右边常数项的 Allowable Increase 和 Allowable Decrease 列表中获得。因此,对于上面的例子,只要 ALIM 的右边常数在[20,120]范围内变化、CLIM 的右边常数大于等于 30 并且 JOINT 的右边常数在[60,160]范围内变化,对偶价格和下降成本的最优值就不变。

注意:在此模型中,所有的约束都有一个用方括号括起来的约束名。如果准备生成一个灵敏度报告,这是一个很重要的习惯。如果没有给约束命名,LINGO 将分配给它们与约束的内部索引相应的名字。在一个最初的模型文本中,这种内部索引不总是和约束的顺序相对应。因此,为了确保灵敏度分析报告右边常数的变化范围意义明确,一定要为所有的约束命名。

如果目标函数中的一个变量是非线性的,它在 Current Coefficient 列表里的值将显示为

NONLINEAR (非线性的)。同理,如果一个约束是非线性的,它在 Current RHS 列表里的值也将显示为 NONLINEAR。

如果系数可以无限增大或减少,则用 INFINITY 表示。

固定变量通常被排除到模型之外,在灵敏度报告中不会出现,含有唯一固定变量的约束也不会在灵敏度报告中出现。例如,将模型中的不等式

```
[ALIM]    A < = 60;
```

改成下列等式:

```
[ALIM]    A = 60;
```

LINGO 现在可以直接解出 A 的值,变量 A 成为固定变量,ALIM 约束成为常量约束。这样,变量 A 不再显示在灵敏度分析报告的 Objective Coefficient Ranges 列表中,并且 ALIM 约束也不会显示在 Righthand Side Ranges 列表中。更新后的灵敏度分析报告如下所示:

```
Ranges in which the basis is unchanged:
     Objective Coefficient Ranges
          Current          Allowable    Allowable
Variable   Coefficient     Increase     Decrease
  C         30.00000       INFINITY     30.00000
       Righthand Side Ranges
  Row      Current      Allowable    Allowable
           RHS          Increase     Decrease
  CLIM     50.00000     INFINITY     20.00000
  JOINT    60.00000     40.00000     60.00000
```

注意:与 Solution 命令相同,LINGO 仅仅将最近一次 LINGO|Solve 命令生成的灵敏度分析报告存放在内存里。如果试图运行 Range 命令而 LINGO 没有相应的灵敏度信息,就会收到一个错误信息。因此,如果打算使用两个或更多的模型且求解时间很长,最好将灵敏度报告都保存在磁盘中,这样以后不需要重新求解模型就可以查阅它们了。

#### 5.3.3.4 LINGO|Generate

Generate 命令(快捷键 Ctrl + G/Ctrl + Q)用于为当前模型创建一个扩展版本。LINGO 将把所有基于集合的紧凑表达式扩展成为等价的完全展开的数量模型。

当运行 LINGO|Generate 命令时,会看到如下两个子菜单:

1) 显示模型(Display model);

2) 不显示模型(Don t display model)。

如果选择 Display model 菜单,LINGO 会将新生成的模型的副本放在一个新的窗口里。在这个新窗口中,可以对模型进行检查、打印或保存到磁盘。如果选择 Don t display model 菜单,LINGO 将生成一个不显示的模型,这个模型将被保存,并在以后可以被适当的求解算法调用。

扩展模型明确地列出了模型里所有的约束和变量。因此,Generate 命令对于寻找潜在的逻辑错误非常有用。

**注意** 如果一个变量在某行中是非线性的，LINGO 将用一个问号代替它的系数值。这对识别模型中的非线性变量也是很有帮助的。

5.3.3.5 LINGO| Picture

Picture 命令（快键键 Ctrl + K）用于以矩阵的形式显示一个模型。在以下两种情况下用矩阵的形式查看模型很有益处。第一，也是最重要的一条，就是非零图片在调试模型中的应用。大多数模型都有很强的重复结构，不正确的模型输入项会在模型的矩阵图中凸现出来。第二，当试图识别模型中的特殊结构时，非零图也很有帮助。例如，如果模型显示成对角线结构，那么将模型分解的算法可能是很有效的。

例如，从 LINGO 的范例中载入 DNRISK. LG4 模型，运行 Picture 命令，将看到图 5-30 所示图片。

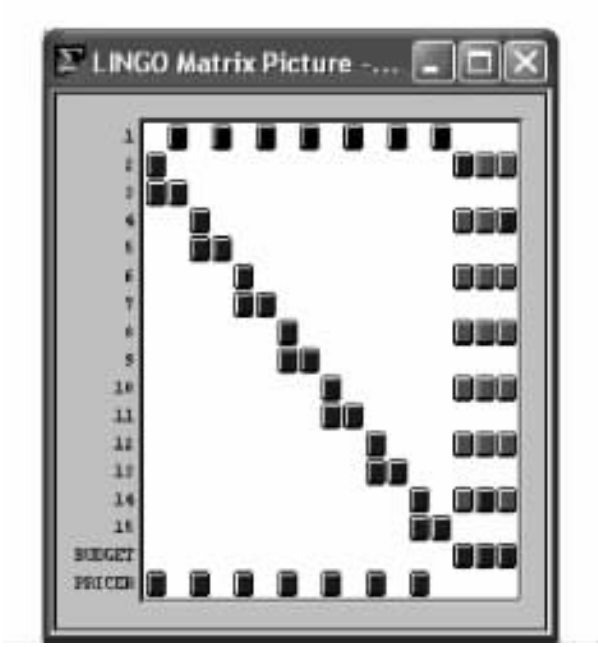



图 5-30 DNRISK. LG4 模型 Matrix Picture

模型中的正系数用蓝色块表示，负系数用红色块表示，各行中的非线性变量用黑色块表示。

放大选择的矩阵区域可以进一步观察矩阵。要做到这一点，首先要将光标放在要观察区域的左上角，按下鼠标左键拖到该区域的右下角，释放左键，LINGO 将放大选择的区域。例如，图 5-31 就是在矩阵中选择 4 × 4 范围放大后的图。

**注意**：只有放大足够的倍数后才能看到实际的系数值、行名和变量名，这时还可以通过移动滚动条来观察整个矩阵。

矩阵视图窗口还有几种交互式特性。访问这些特性，要将光标放在矩阵图上，按下鼠标右键，这时将会出现图 5-32 所示菜单。

各菜单项的功能如下：

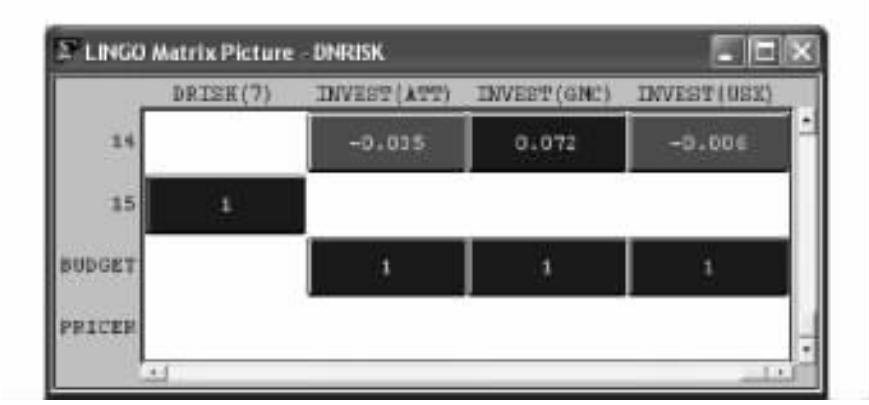



图 5-31　放大的 Matrix Picture

1) Zoom In,以当前光标所在位置为中心放大视图；

2) Zoom Out,以当前光标所在位置为中心缩小视图；

3) View All,查看完整的矩阵视图；

4) Row Names,开关行名的显示；

5) Var Names,开关变量名的显示；

6) Scroll Bars,开关滚动条。


图 5-32　Matrix Picture 中的菜单

目前,矩阵图还不能从 LINGO 中直接打印,但是可以运行 Edit | Copy 命令将矩阵图放到 Windows 剪贴板上,这样矩阵图就可以被粘贴到其他文件中打印。

5.3.3.6　LINGO | Debug

在理想状态下,所有的模型都能返回一个最优解,但总会遇到不可行的或无界的模型。尤其在一个项目开发阶段,还会经常出现一些拼写错误。

在一个大型模型中寻找一个错误,工作量非常大。在不可行解和无界解的线性模型中查找错误,Debug 命令(快捷键 Ctrl + D)是非常有用的。原始模型的一部分将作为错误源被隔离出来,大大缩小查找表达式或数据错误的范围。

Debug 命令能够识别两种集合,即充分集合和必要集合。去掉模型中充分集合的成员对于固定整个模型是充分的。不是所有的模型都有充分集合,在这种情况下,它们将有一个必要集合,从这个集合中去掉任何对象都将固定余下的对象。

假设有一个不可行模型,除了在一行中有错误则是有可行解的,那么这一行将被作为充分集合的一部分列出。如果这个模型有一个必要集合,那么只要它们存在,这个模型就仍然不可行。

以图 5-33 所示的模型为例,第四行中的系数 0.55 本应该是 5.5。当试图求解这个模型时,将得到如图 5-34 所示错误信息。

接下来如果运行 LINGO | Debug 命令,将得到如图 5-35 所示 Debug 报告。

Debug 命令已经正确地识别出错误行 ROW4,当清除这一行后,能充分使整个模型可行。

Debug 命令对于无界解模型有相同的功能。在图 5-36 所示的例子中,引入一个错误 将第

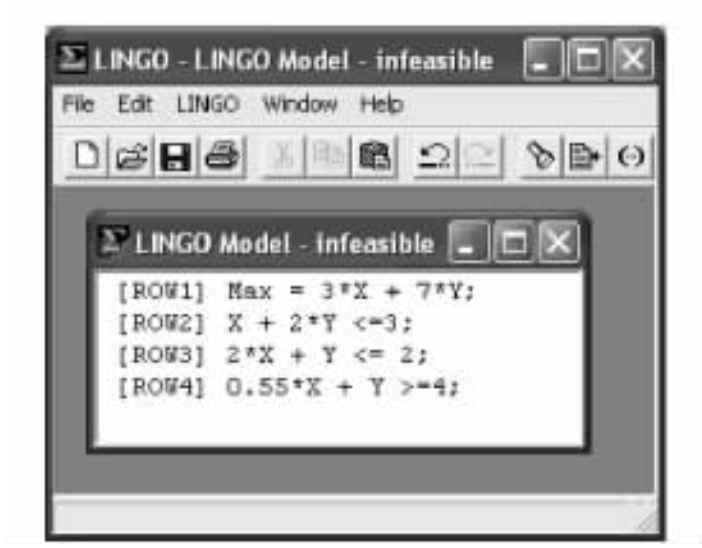



图 5-33　不可行解模型

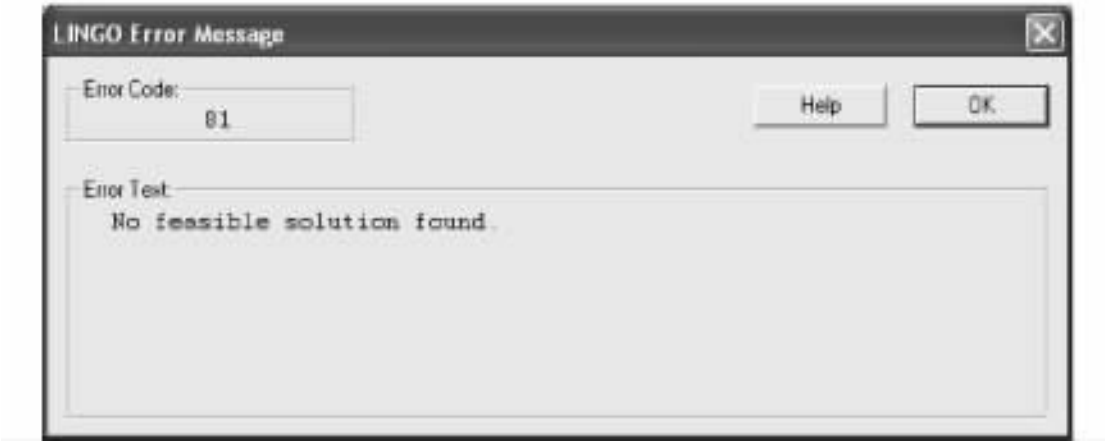



图 5-34　无可行解错误信息

三行ROW3中的变量Z3前的加号改成减号。检查ROW3将发现Z3可以无限增加，从而导致目标值可无限增大。

当运行 LINGO|Solve 命令时，将收到图 5-37 所示的无界解错误信息。

这时，运行 Debug 命令，将看到图 5-38 所示的 Debug 报告。

Debug 命令成功地确定了限制 Z3 对于限制整个模型是充分的。从上面两个例子中可以看出，Debug 命令有可能大大缩小查找错误的时间。

### 5.3.3.7　LINGO|Model Statistics

Model Statistics 命令（快捷键 Ctrl + E）用于为当前模型生成概要统计信息，显示的信息根据模型是否为线性而有所不同。

下面以一个线性运输模型 TRAN.LG4 为例，讨论运行 Model Statistics 命令后生成的信息。

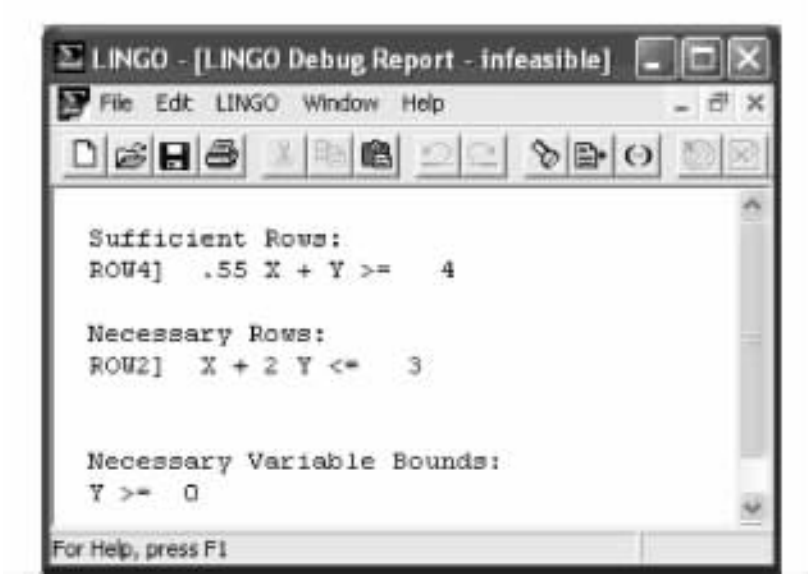



图 5-35 Debug 报告

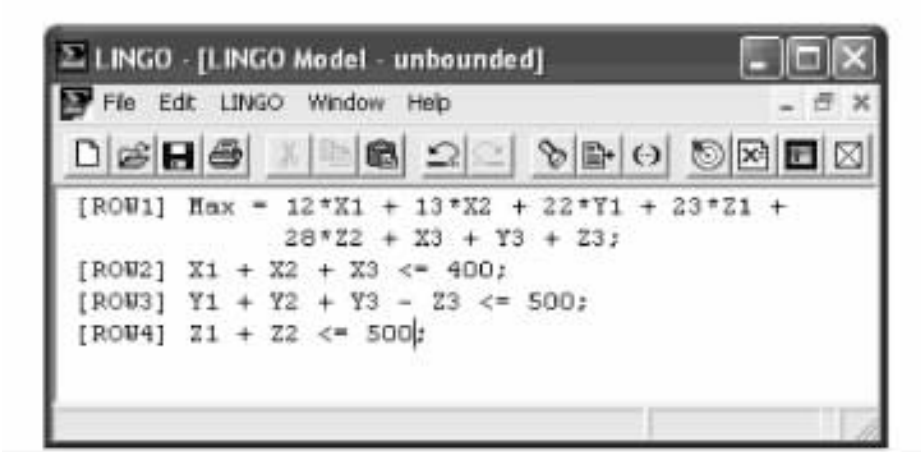



图 5-36 无界解模型

图 5-39 是 Model Statistics 命令为 TRAN.LG4 模型生成的统计报告。该统计报告包含以下五项内容。

第一行包括行数(约束个数)、变量个数(列数)以及整型变量个数。如果模型是线性的,这个报告将指出所有变量是线性的(all are line)。

第二行给出了模型中出现的非零系数的个数。第一个数字显示整个模型中非零系数的总数;Constraint nonz 统计的是所有约束左边非零系数的个数,不包括目标函数中和约束条件右边的非零系数;紧接着显示的是约束条件中为 +1 或 -1 的系数。一般说来,当一个线性模型中 ±1 的系数的个数增加时,该模型更容易求解。最后,LINGO 还给出一个密度数(Density)。该数的定义为:全部的非零系数/[行数 × (列数 - 1)]。对于大型模型来说,密度通常在 0.01 以下。高密度意味着要花很长时间求解模型。

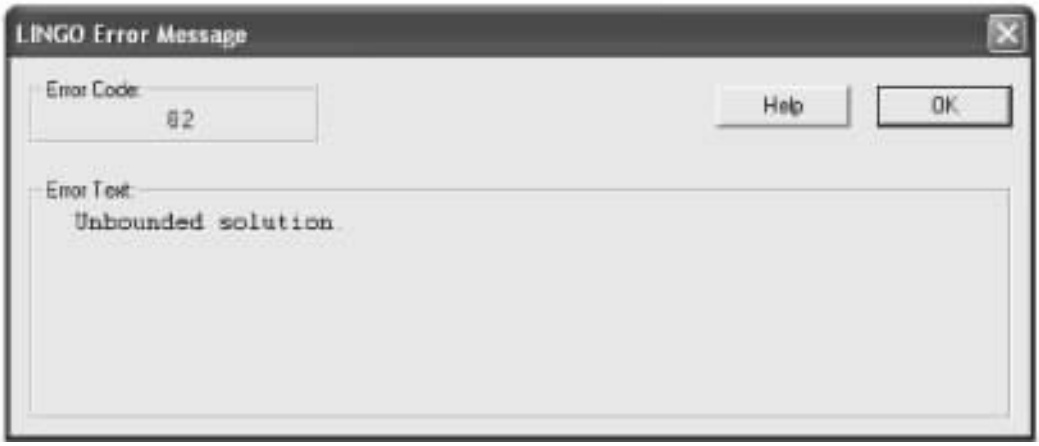



图 5-37　无界解错误信息

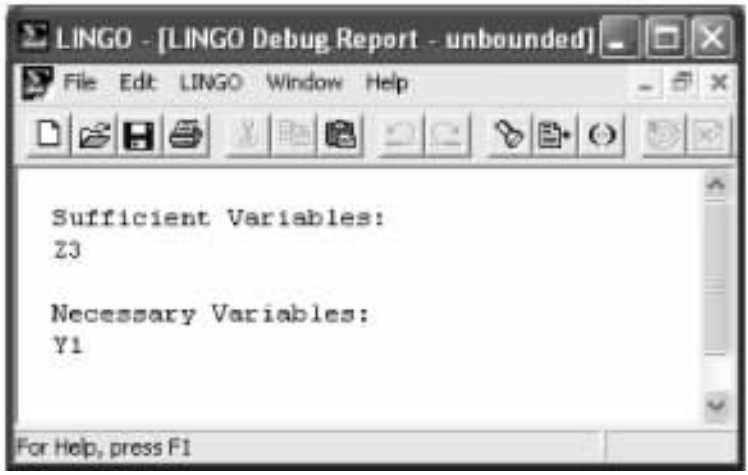



图 5-38　Debug 报告

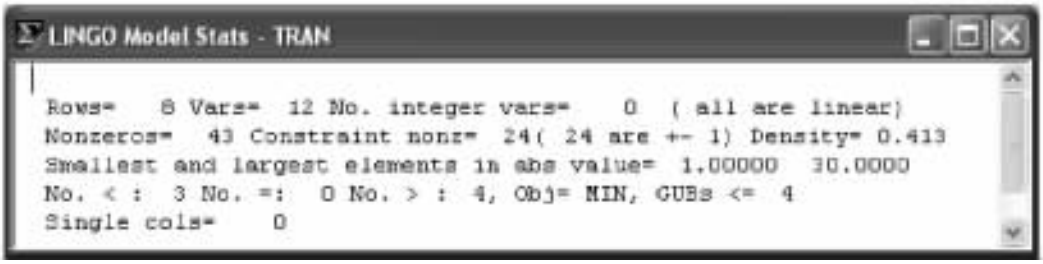



图 5-39　TRAN 模型统计报告

**第三行列出模型中绝对值最大和最小的系数。考虑到模型稳定性，最大系数和最小系数的比值在理想情况下应接近于 1，并且最好保证系数值在 0.000 1 到 100 000 范围内。这个范围外的系数值将增大模型求解的难度。**

第四行按类型（<、=和>）分别列出约束条件的个数、目标函数的意义（即最大 MAX 还是最小 MIN）以及 Generalized Upper Bound（GUB）约束数的上界。GUB 约束是指与模型中其他约束不相关的约束，如果所有的约束都是不相关的，那么 LINGO 将按照每一个约束将模型作为单独的问题进行求解，所以 GUB 统计数是衡量模型简单性的尺度。

第五行列出了仅仅出现在一行中的变量的个数。这样的变量被认为是松弛变量。如果没有在模型中明确增加松弛变量而 Single cols 比零大，则 LINGO 将提示变量名拼写错误。

图 5-40 是 Model Statistics 命令为非线性模型 DNRISK.LG4 生成的报告。

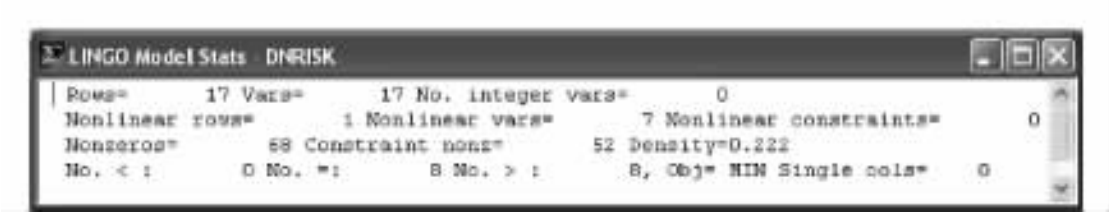



图 5-40 DNRISK 模型统计报告

可见，非线性模型的统计报告中去掉了系数值的范围、正负系数的个数以及 GUB 约束的上界等参数，但在第二行增加了非线性变量个数和非线性的行数。非线性行数包括目标函数，而非线性约束数不包括目标函数。

### 5.3.3.8 LINGO|Look...

Look 命令（快捷键 Ctrl + L）用于生成一个含有模型表达式的报告。在 Look 命令的对话框（图 5-41）中选择 All 或 Selected 可查看相应行的报告。

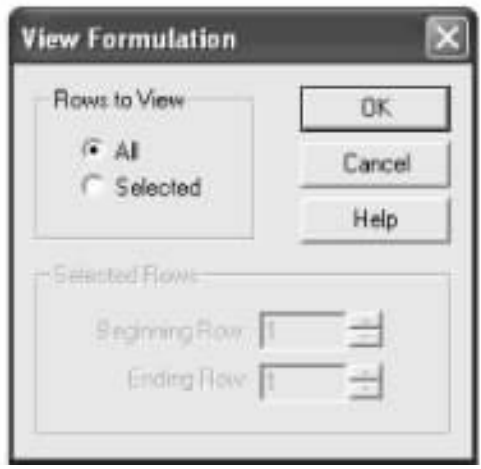



图 5-41 Look 对话框

当选择 Selected 时，Selected Rows 框中用于输入开始和结束行号的文本框将变为可用，这时就可以输入需要显示的行的范围。LINGO 将在新的窗口显示需要的行及行号。

5.3.3.9 LINGO|Options...

Options 命令(快捷键 Ctrl + I)用于改变影响 LINGO 用户界面的参数和 LINGO 求解模型的方法。当运行 Options 命令后,将看到图 5-42 所示对话框。

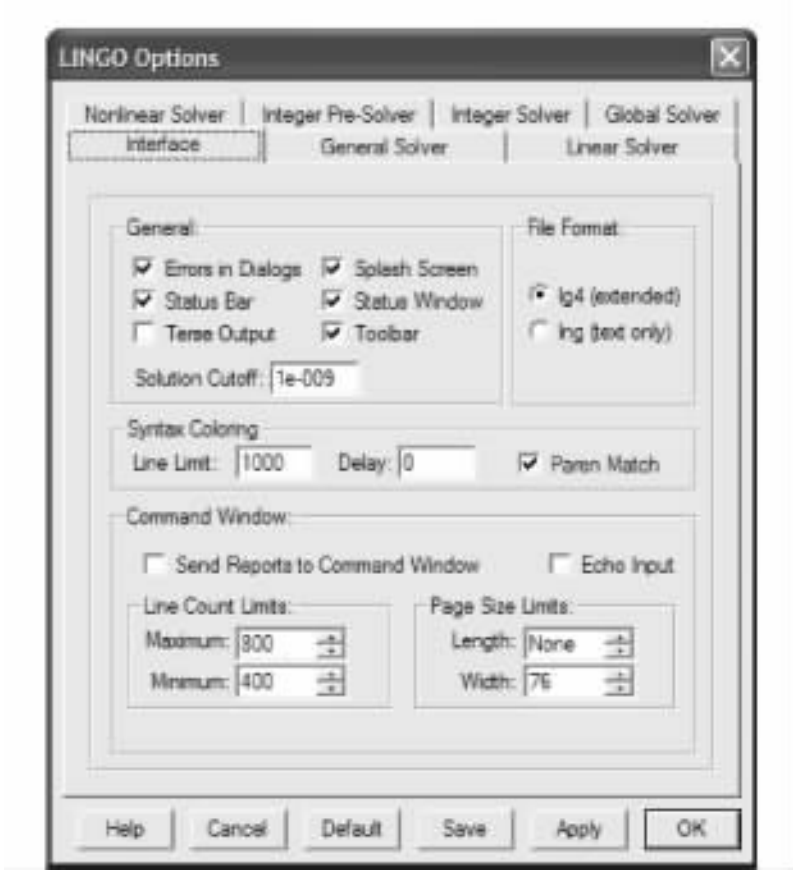



图 5-42 Options 对话框

根据个人的需要设置参数后,单击 Apply 按钮,新的设置就会生效。单击 OK 按钮使新的设置生效的同时关闭对话框。如果希望在以后的 LINGO 程序中应用该设置,就单击 Save 按钮。最初的默认设置在任何时侯都可以通过单击 Default 按钮恢复。

Options 对话框有下列七个页面:

1) Interface(界面);

2) General Solver(一般算法);

3) Linear Solver(线性算法);

4) Nonlinear Solver(非线性算法);

5) Integer Pre-Solver(整数预处理算法);

6) Integer Solver(整数算法);

7) Global Solver(全局算法)。

当第一次运行 Options 命令时,当前页是 Interface 页。Interface 和 General Solver 页包含大多

数用户感兴趣的选项，其他页（Linear Solver、Nonlinear Solver、Integer Pre-Solver 和 Integer Solver）则包含高级用户感兴趣的选项。在下文中将详细介绍每一个页面中的选项。

1. Interface 页

Interface 页的选项对话框如图 5-41 所示。该页中的选项用于控制 LINGO 的外观、输出以及文件的默认格式。

⑴ General

Interface 页中的 General 框包含图 5-43 所示内容。

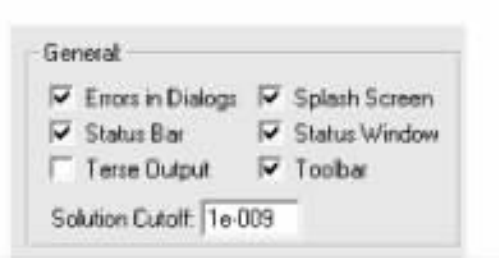



图 5-43　Interface | General 框

选择 Errors in Dialogs（错误信息对话框）复选框，LINGO 将在一个模式对话框中显示求解程序出现的错误信息。只有关闭这个对话框，LINGO 才能进行其他操作。如果不选择这个复选框，LINGO 将把求解的错误信息输送并显示在报告窗口中，还能在没有用户干涉的情况下自动清除该信息。默认状态下求解错误显示在对话框中。

注意 这个选项仅仅可以把 LINGO 求解时产生的错误信息输送到报告窗口，而 LINGO 前后交互出现的错误信息只能显示在对话框中。

选择 Splash Screen（弹出屏幕）复选框，则每次打开 LINGO 时都将显示弹出屏幕。弹出屏幕列出了 LINGO 的版本和版权。

选择 Status Bar（状态栏）复选框，LINGO 将在主窗口的底部显示一个状态栏，其中显示时间、光标位置、菜单提示以及程序的当前状态。Status Bar（清除）复选框，可以将状态栏从屏幕上移除。

选择 Status Window（状态窗口）复选框，则无论何时运行 LINGO | Solve 命令，LINGO 都将显示一个求解状态窗口。默认状态下求解状态窗口是显示的。

选择 Terse Output（扼要输出）复选框，LINGO 将以简要的模式输出模型求解后的报告，而默认状态下 LINGO 的输出是详尽格式的。

选择 Toolbar（工具栏）复选框，LINGO 将显示部分命令的快捷按钮，否则将不显示，默认状态下显示工具栏。


图 5-44　Interface | File Format 框

有时由于舍入的误差，LINGO 求解程序的返回值非常小（小于 $e^{-10}$）。实际上，这些变量的真实值或是零或是小到无关紧要。当研究一个结果报告时这些微小的值可能会分散注意力。Solution Cutoff（结果忽略）参数用于限制一个小的数值，任何小于或等于该值的结果在结果报告中都将显示为零。Solution Cutoff 的默认值为 1e-009。

⑵ File Format

Interface 页中的 File Format 框如图 5-44 所示。

该选项用于控制保存 LINGO 模型时的默认文件格式，有两种不同的格式可供选择，即 LG4 和 LNG。这两种文件格式的特点在前文中已详细介绍。

⑶ Syntax Coloring

Syntax Coloring 框的界面如图 5-45 所示。

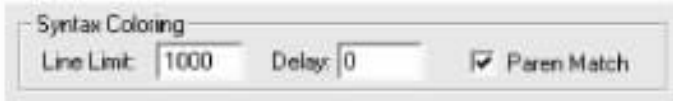



图 5-45　Interface | Syntax Coloring 框

该对话框用于控制 LINGO 编辑器的语法颜色功能。LINGO 编辑器的语法是可以通过颜色识别的。当遇到 LINGO 关键字时,以蓝色显示;注释以绿色显示;剩下的文本则以黑色显示。

如果模型规模很大,那么为语法着色将花费很长时间。Line Limit 域用于设置为语法着色的最大文件规模。行数超过这个参数的模型将不能被设置语法颜色。如果将这个参数值设为 0,那么就没有语法颜色特性,该参数的默认值为 1 000 行。

Delay 值用于设置文本修改后 LINGO 重新对文本进行着色需要等待的秒数。使用较慢机器的用户可以为该参数设置一个较高的值,以避免为语法着色时干扰模型的输入;使用较快机器的用户可以降低这个值,以使文本较快重新着色。该参数的默认值为 0 秒。

如果选择 Paren Match 复选框,当把光标放在某一括号后面时,LINGO 将立即以红色高亮显示与其匹配的另一半括号。并且这些括号一直以红色显示,直到将光标移到其他位置为止,此后括号将重新以黑色显示。默认状态下,该复选框是被选上的。

⑷ Command Window

Interface 页中的 Command Window 框如图 5-46 所示。

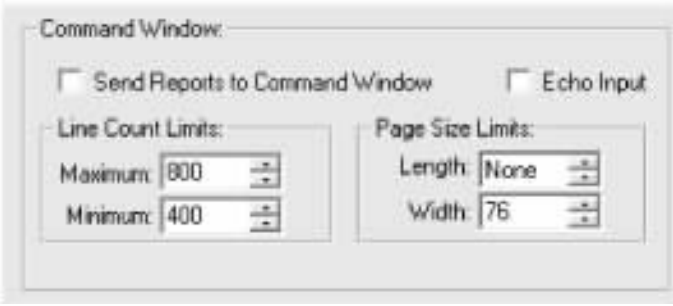



图 5-46　Interface | Command Window 框

该对话框用于自定义 LINGO 命令窗口的用户界面。

LINGO 的命令窗口可以用 Window| Command Window 命令打开。该窗口为用户提供了一个 LINGO 的命令行界面,并且这个界面在 Windows 以外的平台上是一样的。命令窗口对于测试 LINGO 的命令脚本非常有用。

选择 Send Reports to Command Window (输送报告到命令窗口) 复选框,LINGO 将输送任何由其生成的报告到命令窗口,而不是单独的报告窗口。如果希望在一个窗口中包含两个或更多的结果报告,将会用到这个选项。默认值是不输送报告到命令窗口的。

当采用 File | Take 命令运行 LINGO 的命令脚本时,这些命令通常并不显示。如果选择了

Echo Input 复选框，处理的命令将显示在命令窗口中，这对于开发或调试一个命令脚本非常有用。默认状态下该选项是禁止的。

Line Count Limits (行数限制) 用于控制可以保留在命令窗口中的输出行总数。当 LINGO 将输出的内容输送到命令窗口时，这部分内容显示在窗口的底部，而先前的输出将上卷并给新输出腾出空间。Maximum 域用于设置命令窗口中允许的最多行数。当 LINGO 达到此限时，它将从顶部移除一些行，使得余下的行数等于 Minimum 域中的值。如果要保存一个长模型，可以运行 File | Log Output 命令，将所有输出保存到磁盘。Line Count Limits 的默认值最大为 800 行，最小为 400 行。

Page Size Limits (页面大小限制) 用于控制命令窗口的页长和页宽。如果希望在向命令窗口中写入一定数量的行数后暂停，可以通过设置 Page Size Limits 框中的 Length 值实现。当 LINGO 达到这一限制时，在屏幕上将显示如图 5-47 所示界面。


图 5-47 输出暂停界面

LINGO 将一直等待，直到按下 More 按钮后才继续在命令窗口中显示后来的信息。默认值为 None 时没有页长限制。当 LINGO 生成结果报告时，将以一定的宽度限制输出行。在一些报告中，采用换行以使它们适合行宽限，而在另一些报告中，行可能被删节。例如，LINGO 在执行集合运算时连接变量名，则有可能产生较长的变量名 SHIPMENTS (WAREHOUSE1，CUSTOMER2)。如果使用一个特别小的输出宽度，那么结果报告可能被删节。可以通过 Page Size Limits 框中的 Width 域将页宽设置为 64 到 200 之间的任意一个值，默认值为 76。

2. General Solver 页

Options 对话框中的 General Solver 页如图 5-48 所示。

该页面用于控制与 LINGO 求解功能有关的参数。

(1) Generator Memory Limit

General Solver 页中的 Generator Memory Limit 框如图 5-49 所示。

Generator Memory Limit 域用于控制为生成模型留出的内存空间大小。当求解大型模型时，如果留出的内存空间不够，则出现错误信息“The model generator ran out of memory”(模型用尽内存空间)。增加 Generator Memory Limit 域设置的内存数可避免这种错误。值得注意的是，更改此设置后要单击 Save 按钮并重新启动 LINGO 才会生效。

要想准确知道 LINGO 分配给模型生成器的内存大小，可以运行 Help | About 命令。About 对话框将在启动时显示分配给生成器的内存大小。由于分配给模型生成器的内存不能被 LINGO 中的各种求解程序使用，因此，不能分配给生成器过多的内存。

Generator Memory Limit 值的默认大小为 32 MB，如果将该值设置为零，那么 LINGO 将在启动时把所有可使用的内存都分配给生成器。

注意 如果设置的 Generator Memory Limit 值过高，LINGO 和 Windows 将与硬盘反复交换虚拟内存，这将显著减慢计算机速度并导致 LINGO 运行不畅。一般说来，可以设置一个比较大的分配内存以充分处理大型模型，但也不宜过高。可以在任何时候打开求解状态窗口查看 Generator Memory Used 区域中的数值，以确定分配给工作空间的内存数。


**图** 5-48　General Solver **页**

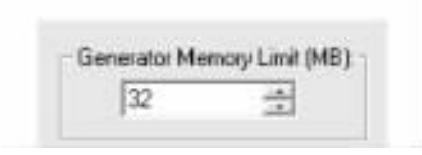



**图** 5-49　General Solver|Generator Memory Limit **框**

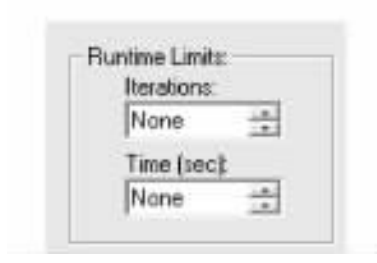



**图** 5-50　General Solver|Runtime Limits **框**

⑵ Runtime Limits

General Solver **页中的** Runtime Limits **框如图** 5-50 **所示。**

**该框中的内容用于控制模型的求解时间。第一项** Iterations **用于设置求解模型时迭代次数的上限。一般说来，较大的模型将花费较长时间进行迭代，非线性模型比线性模型花费时间长。默认的迭代限为** None，**意味着迭代次数没有限制。第二项** Time (sec) **用于设置求解模型时花费时间的上限。默认的时间限是** None，**意味着求解时间没有限制。**

**如果求解模型时达到了以上任何一个限制数，**LINGO **都将返回正常的命令模式。这时如**

果模型含有整型变量，LINGO 将保存目前为止找到的最佳整数解。但是，因为 LINGO 达到时间限后要做大量的工作才能重新设置当前最优解，所以要等待一段时间。

⑶ Dual Computations

General Solver 页中的 Dual Computations 框如图 5-51 所示。


图 5-51 General Solver| Dual Computations 框

该框中的内容用于控制求解模型时运行的对偶计算的级别，可供选择的选项有 None、Prices、Prices and Ranges。当选择 None 时，LINGO 将不计算任何对偶和灵敏度信息。这个选项有利于加快求解速度，但是仅适用于不需要任何对偶信息的情况，而且当关闭对偶计算时，LINGO|Range 命令将失效。当选择 Prices 时，LINGO 将计算对偶值，但不计算对偶值的灵敏度。当选择 Prices and Ranges 时，LINGO 将同时计算对偶值和灵敏度。LINGO 默认的是 Prices 选项。

⑷ Model Regeneration

General Solver 页中的 Model Regeneration 框如图 5-52 所示。

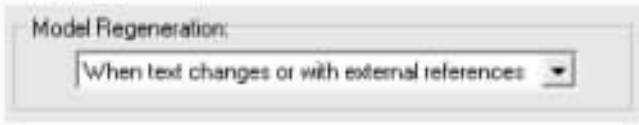



图 5-52 General Solver|Model Regeneration 框

该框中的内容用于控制 LINGO 更新模型的条件。触发 Model Regeneration 的命令有 LINGO|Solve、LINGO|Generate、LINGO|Model Statistics、LINGO|Picture、LINGO|Debug 以及 File|Export File 等。

下拉菜单里可供选择的选项有：

Only when text changes——模型生成以后，仅仅当模型中的文本发生改变时 LINGO 才更新模型；

When text changes or with external references——（默认）无论是模型中的文本还是模型中涉及的外部数据源（例如文本文件、数据库或电子表格）被改变时，LINGO 都将更新模型；

Always——每次需要生成模型信息时，LINGO 都将更新一次模型。

⑸ Linearization

General Solver 页中的 Linearization 框如图 5-53 所示。

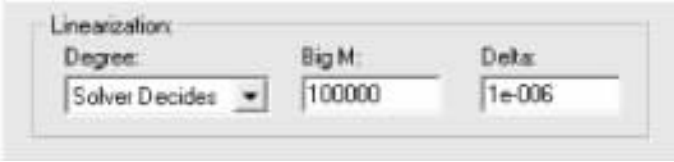



图 5-53 General Solver|Linearization 框

该框中的内容用于控制 LINGO 的线性化选项。许多非线性运算可以用等价的线性运算

代替，其最终目标是将模型中所有的非线性运算用等价的线性运算代替，从而可以用速度较快的线性算法求解。这个过程被称为线性化过程。

Degree 决定 LINGO 试图线性化模型的程度，可使用的选项有 Solver Decides、None、Low 和 High。选择 None，LINGO 不进行线性化。选择 Low，LINGO 将线性化任何涉及二进制和连续变量的函数，如 @ABS()、@MAX()、@MIN()、@SMAX() 和 @SMIN() 等。High 与 Low 相似，但 High 增加了对所有逻辑运算（#LE#、#EQ#、#GE# 和 #NE#）的线性化。选择 Solver Decides 时，如果变量数小于等于 12，LINGO 将最大限度地将模型线性化，否则 LINGO 将不执行任何线性化操作。默认状态下，LINGO 选择 Solver Decides 选项。

Delta 系数用于设定所希望的作为线性化过程增加的约束条件的接近程度。大多数模型不需要改变这个参数，但是稍增加 Delta 值，一些数学表达式就可以受益。LINGO 默认的 Delta 值为 1e-006。

当 LINGO 线性化一个模型时，将向用于优化模型的数学程序中增加强制约束，这些强制约束的格式如下：

```
f(Adjustable Cells) = M*y
```

其中 M 是 Big M 系数，y 是 0/1 变量。如果变量发生一些变化，强制约束会驱使 y 取 1。这时，如果设置的 Big M 值太小，模型可能最终无可行解。因此，将 Big M 值设置成比较大的值可以减少模型不可行的机会。但是 Big M 值设置得较大时，会导致产生算法的数值稳定性问题。

(6) Allow unrestricted use of primitive set member names

General Solver 页中的 Allow unrestricted use of primitive set member names（允许自由使用基本集合元素名）复选框使得模型与 LINGO 早期版本的模型兼容。

在许多情况下，需要知道基本集合元素在集合中的索引。LINGO 4.0 以前的版本，在模型等式中直接用基本集合中的元素名就可以得到索引值。但是当从外部源导入集合元素时可能出现问题，因为这种情况下，不需要预先知道集合元素名，所以当导入的基本集合元素名与模型中的变量名恰好重合时将会发生意外的结果。可以肯定的是，LINGO 不会将其当作可优化的变量，而是将其视为等于基本集合元素索引值的常量。因此，从 LINGO 4.0 开始，运行下面的模型将会出错。

```
MODEL:
SETS:
  DAYS /MO TU WE TH FR SA SU/;
ENDSETS
  INDEX_OF_FRIDAY = FR;
END
```

如果需要集合 DAYS 里 FR 的索引，应该使用 @INDEX 函数：

```
INDEX_OF_FRIDAY = @INDEX(DAYS, FR);
```

默认情况下，LINGO 不选择此复选框。

(7) Check for duplicate names in data and model

General Solver 页中的 Check for duplicate names in data and model 复选框用于检测旧版本的

LINGO 模型,例如基本集合元素在模型中出现的位置。如果集合元素名或变量名在模型中重复,在下一次运行模型时 LINGO 将产生错误信息。默认状态下,LINGO 不选择该复选框。

⑻ Use R/C format names for MPS I/O

General Solver 页中的 Use R/C format names for MPS I/O 复选框使 LINGO 在执行 MPS 文件格式的输入输出时,将所有的变量名和行名转换成 R/C 符号。默认状态下,LINGO 不使用 R/C 格式名。

3. Linear Solver 页

Options 对话框中的 Linear Solver 页如图 5-54 所示。

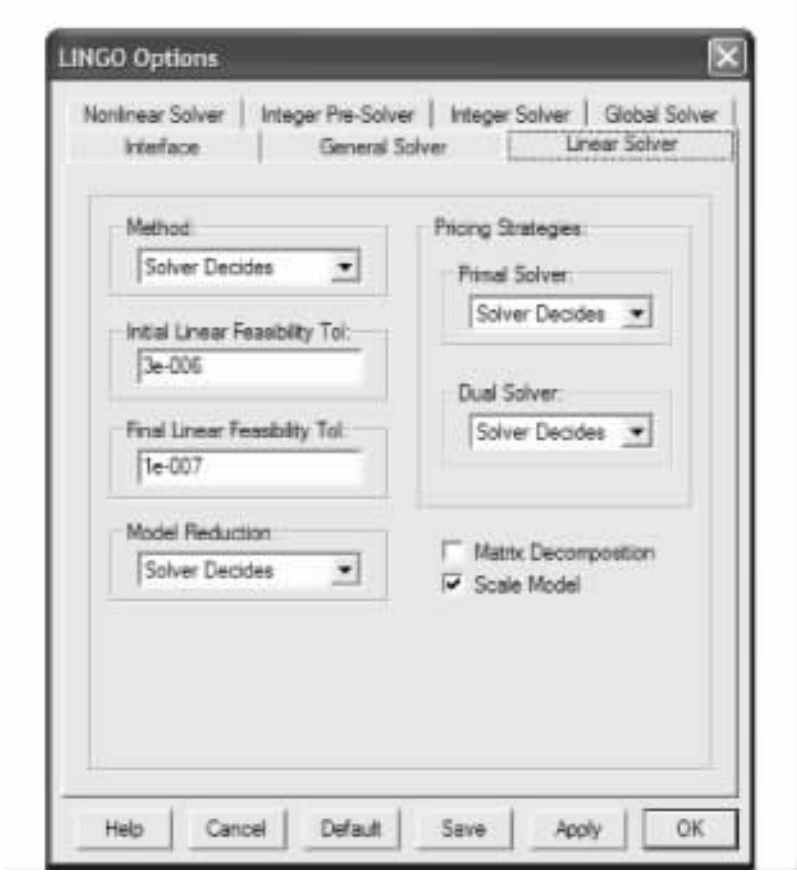



图 5-54 Linear Solver 页

该页面控制的选项用于配合 LINGO 中的线性算法。线性算法作为分支定界过程中的一部分,用于求解线性模型和混合整数线性模型。

⑴ Method

Linear Solver 页中的 Method 框用于控制 LINGO 求解器采用何种算法。下拉菜单中可供选择的选项有:

Solver Decides——自动选择最适合的算法;

Primal Simplex——运用原始单纯形算法;

Dual Simplex——运用对偶单纯形算法;

Barrier——运用障碍算法(即内部点算法)。

单纯形算法沿可行域外边缘移动至最优解,而内部点算法(Interior point)即 Barrier 算法,在可行域内部移动。一般说来,对一个特定的模型很难确定哪一种算法更快。大体的结论是:求解行数比列数少的稀疏模型时用原始单纯形算法较好,求解列数比行数少的稀疏模型时用对偶单纯形算法较好,而求解密集型结构的模型或大型模型时用 Barrier 算法较好。

Barrier 算法在 LINGO 软件包的附加选项中才能得到。此外,如果模型中有整型变量,Barrier 算法会在分支定界树的根节点处求解线性规划模型,但是在以后的节点上不一定使用。从性能的观点看,Barrier 算法对整数模型的作用较小。LINGO 默认的选项是 Solver Decides。

(2) Initial Linear Feasibility Tol 和 Final Linear Feasibility Tol

Linear Solver 页中的 Initial Linear Feasibility Tol 框和 Final Linear Feasibility Tol 框用于控制线性算法的可行性容差。这些容差值控制线性模型的约束条件被满足的程度。

由于计算机的浮点计算精度有限,LINGO 不能总是严格地满足每一个约束。因此,LINGO 用这两个容差作为约束允许的容差限度,只要在这个限度内,就可以认为约束得到满足。这两个容差分别称为 Initial Linear Feasibility Tolerance (ILFT)和 Final Linear Feasibility Tolerance (FLFT),默认值分别为 0.000 003 和 0.000 000 1。

当算法开始迭代时首先用到 ILFT 容差。在求解过程的早期阶段,算法为了加快求解速度而较少关心求解精度。当 LINGO 认为有一个最优解时,它将转向限制性更强的 FLFT 容差。在这个阶段,需要一个高的精确度。因此,FLFT 容差应比 ILFT 容差小。

当 LINGO 返回一个几乎可行的解时,将用到这两个容差,这点可以通过检验结果报告中的 Slack or Surplus 列的值验证。如果这一列中仅有几行是小的负数值,说明该解已经接近可行。放松(增大)ILFT 和 FLFT 的值对于找到一个可行解很有帮助,尤其是在很难放缩的模型(在同一个模型中同时有很大和很小的系数)和一些约束的度量单位非常小甚至可以忽略的模型中。例如,对一个数百万元标准的预算限制,几美分的容差不会影响结果。当缺少重新放缩模型的首选方法时,放松可行性容差是非常有利的。

(3) Model Reduction

Linear Solver 页中的 Model Reduction 框用于控制 LINGO 线性算法执行时是否对模型进行减量(reduction)操作,选项有:

Off——使减量失效;

On——对所有模型进行减量操作;

Solver Decides——LINGO 决定是否采用减量操作。

当减量选项被激活时,LINGO 将在模型求解前尝试从表达式中识别并移除无关的变量和约束。在一定情况下,这将大大缩减最终求解模型的大小。然而,有时减量操作仅仅增加了求解时间,却没有缩减模型大小。LINGO 默认的是 Solver Decides 选项。

(4) Pricing Strategies

Linear Solver 页中的 Pricing Strategies 框如图 5-55 所示。

该框中的内容用于控制 LINGO 单纯形算法应用的定价策略,定价方式决定了单纯形算法运算过程中出基变量的策略。

Primal Solver 框中有以下选项:

Solver Decides——LINGO 选择它认为最合适的定价方法；

Partial——每一次迭代都对一个小的变量子集进行定价并且间歇地给所有的变量定价，以确定一个新的变量子集；

Devex——每一次迭代都采用一个最陡边缘 (steepest-edge) 逼近法给所有的列定价。

Partial 定价法能够提供较快的迭代速度，而 Devex 定价法较慢，并且导致较少的全局迭代，但对于退化的模型很有用。因此，很难预先知道哪一种方法较好。


图 5-55 Linear Solver | Pricing Strategies 框

Dual Solver 框有以下选项：

Solver Decides——LINGO 选择它认为最合适的定价方法；

Dantzig——不管新引进的变量每移动一个单位时其他变量必须移动多远，对偶单纯形算法都将趋于选择那些对于改进目标值能提供最高绝对速率的变量；

Steepest-Edge——对偶算法将花较多的时间选择变量，选择变量的方法是调整某个特殊变量时观察目标值的总的改善程度。

Dantzig 定价法一般会产生较快的迭代速度，模型中的其他变量可能很快落入使目标值增长较少的范围内。而选择 Steepest-Edge 选项，每一次迭代都将使目标值增长较大，从而导致较少的全局迭代。但是，Steepest-Edge 法的每一次迭代都将花费较长的时间。

在 Primal Solver 和 Dual Solver 框中 LINGO 默认的都是 Solver Decides 选项。

(5) Matrix Decomposition

Linear Solver 页中的 Matrix Decomposition 复选框用于分解矩阵。许多大规模线性和混合型整数问题都有约束矩阵，这些矩阵可以完全分解成一系列的子块。如果可以完全分解，LINGO 将依次求解这些独立的问题并为原始的模型生成结果报告，这样就大幅度提高了求解速度。LINGO 默认的是不采用矩阵分解。

(6) Scale Model

Linear Solver 页中的 Scale Model 复选框用于确定是否重新调整矩阵的系数，以使最大和最小系数的比率降低。这将会降低舍入错误的机会，在线性算法中还会提高数学稳定性和精确度。LINGO 默认的是运用放缩模型功能。

4. Nonlinear Solver 页

Options 对话框中的 Nonlinear Solver 页如图 5-56 所示。该页面控制的选项用于配合 LINGO 中的非线性算法。

(1) Initial Nonl Feasibility Tol 和 Final Nonl Feasibility Tol

Nonlinear Solver 页中的 Initial Nonl Feasibility Tol 和 Final Nonl Feasibility Tol 框用于控制非线性算法采用的可行性容差，控制方法与前面所述的线性算法相同，这里不再赘述。这两个容差的默认值分别为 0.001 和 1e-006。

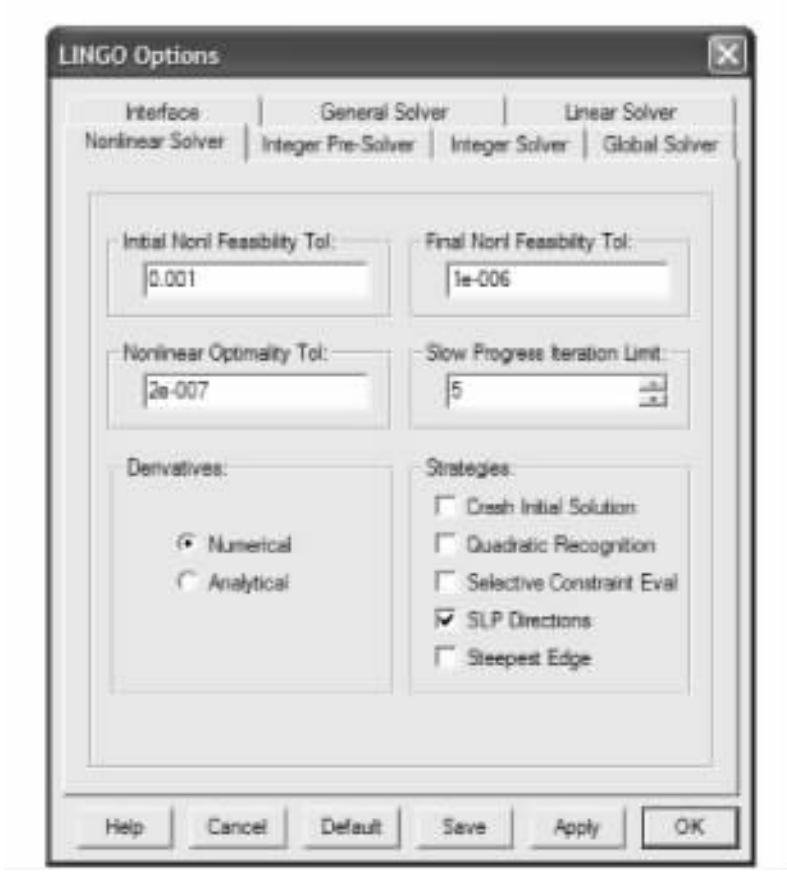



**图** 5-56　Nonlinear Solver **页**

⑵ Nonlinear Optimality Tol

Nonlinear Solver **页中的** Nonlinear Optimality Tol **框用于控制变量的调节。当求解模型时，非线性算法会不断地计算梯度。梯度表示变量微小的变化所引起目标函数改进的速度。如果一个给定变量的改进梯度小于等于非线性优化容差值** (Nonlinear Optimality Tol)，**那么进一步调整变量值没有什么用处。朝零的方向降低此容差值将使求解时间增长，但可以使表达较差的模型或不能很好缩放的模型得到较好的解。** Nonlinear Optimality Tol **的默认值为** 2e-007。

⑶ Slow Progress Iteration Limit

Nonlinear Solver **页中的** Slow Progress Iteration Limit (SPIL) **框用于控制在目标值有很少或没有进展时终止求解过程。具体地讲，如果目标函数值在迭代** n **次后仍没有显著进展，非线性算法将终止求解过程，其中** n **为** SPIL **值。虽然增加此容差值会使算法的运行时间延长，但这对围绕最优解周围目标函数值相对平坦的模型很有帮助。** SPIL **的默认值为** 5。

⑷ Derivatives

Nonlinear Solver **页中的** Derivatives **框用于控制计算非线性模型导数的方法。有两种不同的方法可供选择，即数值法** (Numerical) **和解析法** (Analytical)。**数值法是利用有限差分法求导，而解析法则是通过对约束中的数学运算进行直接符号解析求导。**

**不是所有的函数都有解析导数，**LINGO **现在支持解析法求导的数学运算包括加、减、乘、除**

和乘方。如果一个约束不能用解析法求导时，LINGO 将使用数值法求导。用解析法求导能提高一些非线性模型的求解速度和精度。LINGO 默认的是 Numerical 选项。

(5) Strategies

Nonlinear Solver 页中的 Strategies 框如图 5-57 所示。


图 5-57 Nonlinear Solver | Strategies 框

若选择 Crash Initial Solution 复选框，当求解模型时，LINGO 的非线性算法将启发式地生成一个“好”的出发点。如果这个出发点确实是好的，那么算法接下来的迭代将缩短整个运行时间。默认状态下，LINGO 不选 Crash Initial Solution 复选框。

选择 Quadratic Recognition 复选框，LINGO 将用代数预处理的方法确定任意一个非线性模型是否是二次规划模型 (Quadratic Programming，QP)。如果一个模型被认为是 QP 模型，那么它将被传递到较快的二次算法中求解。值得注意的是，QP 算法并不包含在 LINGO 的标准基础版本中，而是作为屏蔽选项的一部分出现。默认状态下，LINGO 不选择 Quadratic Recognition 复选框。

选择 Selective Constraint Eval 复选框，LINGO 的非线性算法将仅仅对基本必要约束进行估值。因此，不是所有约束在每次迭代时都被估值，这将缩短求解时间。但是，如果模型含有在这些区域内未定义的函数时将会出现错误。如果发现有的约束移进无定义区间，则 LINGO 在多次迭代过程中不会对该约束估值。在这种情况下，算法不可能返回一个有效点，并且求解过程以错误终止。关掉 Selective Constraint Eval 将排除这些错误。默认状态下，LINGO 默认不选择 Selective Constraint Eval 复选框。

选择 SLP Directions 复选框，LINGO 的非线性算法将用连续线性规划 (Successive Linear Programming，SLP) 估计新的查找方向。在查找过程中运用线性逼近的方法加快迭代时间。然而，一般地，当 SLP Directions 被应用时，总的迭代次数将趋于增加。默认状态下，LINGO 选择 SLP Directions 复选框。

选择 Steepest Edge 复选框，LINGO 的非线性算法就会运用 Steepest-Edge 策略选择参加迭代的变量。当模型没有在 Steepest Edge 模式时，不管新引进的变量每单位移动其他变量必须移动多远，非线性算法都将趋于选择那些能够为改进目标值提供最高绝对速率的变量。而选择 Steepest Edge 选项后，非线性算法将花费很长时间选择变量，选择变量的方法是观察其他非零变量引起的目标函数的增长速率的大小。因此，平均地说，每一次迭代都将引起目标值较大的增长。一般地，选择 Steepest Edge 选项会减少迭代次数，但每次迭代将花费较长时间。默认状态下，LINGO 不选择 Steepest Edge 复选框。

5. Integer Pre-Solver 页

Options 对话框中的 Integer Pre-Solver 页如图 5-58 所示。

该页面控制的选项用于配合 LINGO 的整数预处理 (Integer Pre-Solver) 算法。整数预处理算法用于完成模型的再生成工作，使得传送到分支定界算法的最终表达式能够以最快的速度求解。重新生成的模型等价于原始表达式，但是构成方式发生了变化。这种方式最适合于用分


图 5-58　Integer Pre-Solver 页

支定界整数规划算法求解。

整数预处理算法仅适合于线性整数模型（也就是用 @BIN 函数和 @GIN 函数限定一个或更多变量为整型变量的模型）。因此，该页的设置对非线性整数模型没有影响。

⑴ Heuristics

Integer Pre-Solver 页中的 Heuristics 框用于控制整数算法应用的整数规划的试探水平。在分支定界树每个节点采用连续解试探，尝试找到一个好的整数解。试探法仅适用于线性的整数规划模型，在非线性模型或纯线性规划中没有意义。

Level 域用于控制试探的层次和数量，可以从 0 到 100。Min Seconds 域用于指定每个节点试探花费的最小时间。Level 的默认值是 3，Min Seconds 的默认值是 0。

⑵ Probing Level

Integer Pre-Solver 页中的 Probing Level 选项用于在混合整数线性模型中执行探测操作。探测包括靠近模型中的整型变量，推导出范围更小的变量限制和右边的常数值。在许多情况下，探测可以充分紧缩模型并缩短整个求解时间。但是在另一些情况下，探测并不能使模型紧缩太多，反而由于花费额外的探测时间而使总的求解时间增长。

Probing Level 下拉菜单如图 5-59 所示。Probing Level 域中的选项表示不同的探测水平。探测水平 1 表示探测功能无效，从 2 到 7 探测水平依次升高。这个选项的默认设置为 Solver De-

cide，即由 LINGO 决定探测水平。

(3) Constraint Cuts

Constraint Cuts 框中的 Application 下拉菜单用于控制结果树中的节点，在这些节点处分支定界算法将增加剪切量。Application 下拉菜单包含 Root Only、All Nodes 和 Solver Decides 三个选项。

选择 Root Only 选项，算法仅仅在结果树的第一个节点（即根结点）处附加剪切量。选择 All Nodes 选项，算法将在树中每个节点处都附加剪切量。选择 Solver Decides 选项，将由算法动态决定何时在哪个节点附加剪切量，这也是默认的选项。

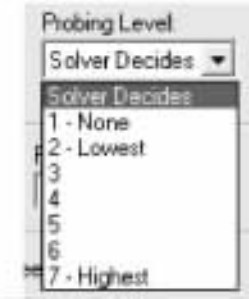



图 5-59 Integer Pre-solver | Probing Level 选项

Constraints Cuts 框中的 Relative Limit 域用于控制约束剪切量的大小，此剪切量由预处理算法产生。附加约束剪切量有利于求解大多数整数规划模型，但是，附加剪切量将使生成结果的时间远远超过保存结果的时间。因此，LINGO 增加了该选项以限定生成的约束剪切量大小。该值的默认设置为 0.75，即为原始表达式中真实约束的 0.75 倍。

Constraint Cuts 框中的 Max Passes 域用于控制整数预处理算法求解模型时的迭代次数，从而决定表达式的最佳切口。一般说来，每一次连续传递收益都会下降。在有些点处，附加传递只会增加总求解时间，因此，LINGO 将强加一个最大传递限数。默认分支树根节点处传递次数限为 200 次，所有其后的节点传递次数限为 2 次。可以在 Root 和 Tree 区域中修改这些值。

Constraint Cuts 框中的 Types 框用于选择 LINGO 生成约束剪切量的不同策略。LINGO 可用 12 种不同策略生成约束剪切量。默认情况下采用除 Basis 之外的所有的剪切量生成策略。

关于不同策略的详情已超出了本书的范围。感兴趣的读者可以查阅一些有关整数规划技术的书籍。

6. Integer Solver 页

Options 对话框中的 Integer Solver 页如图 5-60 所示。该页面中的设定值用于配合求解整数模型的分支定界算法的操作。

(1) Branching

Integer Solver 页中的 Branching 框包含 Direction 和 Priority 两个选项，用于控制 LINGO 分支定界算法应用的分支策略。

当求解整数规划模型时，LINGO 将使用分支定界算法。分支定界算法中的基本操作之一就是变量分支。分支过程包括驱使一个当前为小数的整数变量向下一个最大或最小整数值变化。例如，有一个普通整型变量，当前值为 5.6。如果 LINGO 将这一变量进行分支操作，那么必须选择设置该变量初值是 6 还是 5，Direction 选项就用于控制 LINGO 做出哪种分支决定。Direction 选项的下拉列表如图 5-61 所示。

默认选项为 Both。LINGO 将根据明智的推测确定是首先向上还是向下分解每一个单独变量。如果选择 Up，LINGO 总是首先向上分解变量 如果选择 Down，LINGO 总是首先向下分解变量。在许多情况下，Both 选项会得到最好的结果。

Priority 域用于控制 LINGO 首先分解何种整型变量，下拉列表如图 5-62 所示。

选择 Binary 选项，LINGO 将把分解优先权给二进制变量。选择 LINGO Decides 选项，LINGO


**图** 5-60　Integer Solver **页**

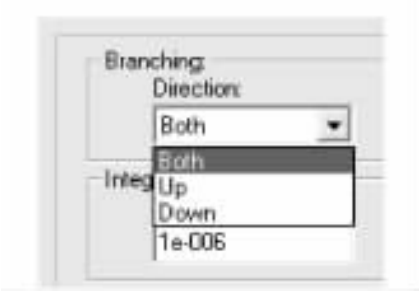



**图** 5-61　Integer Solver | Branching | Direction **下拉列表**

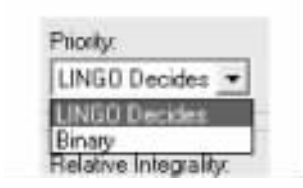



**图** 5-62　Integer solver | Branching | Priority **下拉列表**

**则会基于一个明智的推测选择下一个用于分解的整型变量，不管这个变量是二进制的还是普通的，默认选项是**LINGO Decides**，这样通常能得到最好的结果。**

⑵ Integrality

**由于计算机潜在的舍入误差，**LINGO **总是不能为每个变量找到精确的整数解。**Integer Solver **页中的** Integrality **框包含** Absolute Integrality **和** Relative Integrality **两个选项，用于控制能接受的取整容差量。**

绝对取整(Absolute Integrality)容差指可以接受的绝对取整容差量。例如,如果 X 是一个整数变量,I 是最接近 X 的整数,那么如果

$$|X - I| \leqslant \text{绝对取整容差}$$

则 X 将接受这个整数值。绝对取整容差的默认值为 1e-006。尽管可以将这个容差设置为 0,但是这样做的结果将使有可行解的模型成为无可行解模型。

相对取整(Relative Integrality)容差指可以接受的相对取整容差量。例如,设 I 是最接近 X 的整数值,如果

$$\frac{|X - I|}{|X|} \leqslant \text{相对取整容差}$$

则 X 将接受该整数值。相对取整容差的默认值为 8e-006。

⑶ LP Solver

在一个混合线性整数规划模型中,LINGO 的分支定界算法将在结果树的每个节点处求解线性规划模型。LINGO 可以选择 Primal Simplex、Dual Simplex 或 Barrier 算法来处理这些线性模型。Integer Solver 页中的 LP Solver 框包含 Warm Start 和 Cold Start 两个选项,用于控制线性规划算法的选择。

当前一节点的解存在时,用 Warm Start 选项决定启用何种线性算法,并且该算法被分支定界算法应用在结果树中每一个节点处。当前一节点的解不存在时,用 Cold Start 选项决定启用何种线性算法。

Warm Start 域的下拉列表如图 5-63 所示。一般说来,LINGO Decides 将产生最好的结果。Barrier 算法不能充分利用已有的结果,所以一般不能得出较好的结果。分支优化时 Dual 比 Primal 速度要快。


图 5-63 Integer solver|LP Solver|Warm Start 下拉列表

Cold Start 选项的用法与 Warm Start 相同。

⑷ Optimality

Integer Solver 页中的 Optimality 框用于确定 Absolute、Relative 和 Time to Relative 三个容差值。这些容差值表示期望的求解结果与最优解的接近程度。理想状态下,总是希望算法能找到模型的最优解。但是,整数规划问题非常复杂,并且寻找绝对最优解的额外计算量也非常大,所以求解一个大型的整数模型,通常选择运行几分钟求得一个与真正最优解值相差几个百分点的解,而不是运行几天时间以求得真正最优解,这时这几个容差值非常有用。

Absolute Optimality 值是一个正值 r,指示分支定界算法只寻找比目前找到的最优整数解的目标函数值至少好 r 个单位的整数解。在许多整数规划模型中,大量分支有大致相同的潜力。这个容差有利于防止分支定界算法被这些分支干扰,因为这些分支并不能提供一个比当前解显著好的解。

一般说来,不必设置 Absolute Optimality 容差值。对于一个表达不理想的模型,可能需要稍增加此容差以提高求解性能。在许多情况下,要想提高求解性能,应尝试使用 Relative Optimality 容差值而不是 Absolute Optimality。Absolute Optimality 的默认值为 8e-008。

Relative Optimality 值 r 的范围从 0 到 1,指示分支定界算法仅仅寻找能使目标函数值至少

比目前找到的最优整数解的目标函数值改善 r × 100% 的整数解。

用这种方法修改查找程序的最终结果有积极的一面，因为能缩短求解时间；也有消极的一面，因为最终获得的最优解可能并不是真正的最优解。但是，可以保证这个解在真正最优解的 r × 100% 范围以内。

Relative Optimality 的典型值在 0.01 ~ 0.05。换句话说，如果得到一个与真正最优解之差在 1% ~ 5% 的解是令人满意的。Relative Optimality 的默认值为 5e-008。

如果一个整数模型是相对容易求解的，则希望直接求真正的最优解，而不用 Relative Optimality 容差。另一方面，如果运行一段时间后，最优解不能立即求得，那么就可能希望算法应用 Relative Optimality 容差。此时，可以用 Time to Relative 容差值。该容差值是分支定界算法开始应用 Relative Optimality 前的时间限。在起初 n 秒内（此处 n 为 Time to Relative 容差值），分支定界算法不用 Relative Optimality 容差并试图找到模型真正的最优解；n 秒后，算法将使用 Relative Optimality 容差并继续寻找最优解。

(5) Tolerances

Integer Solver 页中的 Tolerances 框包含三个容差设定值，分别是 Hurdle、Node Selection 和 Strong Branch。这些容差设定值用于控制分支定界算法求解整数规划模型的分支策略。

如果知道一个模型解的目标值，可以将其输入到 Hurdle 域内。这个值会使分支定界算法缩小最优解的搜索范围，也就是说，LINGO 将只查找比阈值（Hurdle）好的目标函数值对应的整数解。当 LINGO 开始寻找整数解时，这一容差值开始有效。如果目标值劣于该阈值，LINGO 在查找树中将忽略这些分支，因为在其他一些分支中存在一个比较好的解（目标值等于阈值的解）。因此，一个好的阈值可以大大缩短求解时间。一旦 LINGO 找到一个初始整数解，Hurdle 值就不再有效。这时，Relative Optimality Tolerance 开始生效。

注意：当输入一个阈值时应确保存在一个解，这个解的目标值至少和阈值一样好。如果这样的解不存在，LINGO 将不能找到模型的可行解。

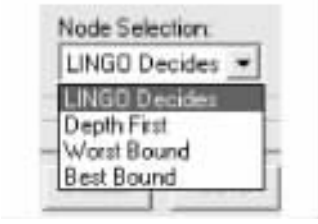



图 5-64 Integer solver | Tolerances | Node Selection 下拉列表

分支定界算法能自由地决定怎样生成一个分支结果树。Node Selection 选项用于控制算法在树中选择分支节点的次序。Node Selection 下拉列表如图 5-64 所示，有如下四个选项可供选择：

LINGO Decides——默认选项，LINGO 根据分析选择最佳的节点用于分支；

Depth First——采用深度优先策略生成分支树；

Worst Bound——挑选具有最弱限制的节点；

Best Bound——挑选具有最强限制的节点。

一般说来，LINGO Decides 能得出最好的结果。

Strong Branch 选项用于在分支树的头 n 个层中运用更精深的分支策略。这里 n 为 Strong Branch 的值。在这最初的几层中，LINGO 精选一个小数变量子集作为分支候选，然后在子集中的每一个变量上进行试探性分支，选择能使目标值改进最大的变量作为最终的分支候选。尽管 Strong Branch 对于快速紧缩边界很有用，但是它将增加求解时间。因此，可以尝试不同的设置以决定哪种设置对建立的模型最合适。Strong Branch 的默认值为 10。

7. Global Solver 页

Options 对话框中的 Global Solver 页如图 5-65 所示。该页中的内容用于控制 LINGO 全局算法功能的运行。全局算法工具箱是 LINGO 的附加选项，必须明确购买全局算法选项才能使用它的功能。


图 5-65 Global Solver 页

LINGO 利用线性模型凸的特性可以找到全局最优解，但是模型多数是非线性的。LINGO 默认的 NLP 算法只进行局部搜索，这可能导致 LINGO 在局部最优点上停止，而错过全局最优点。全局算法工具箱具有通过局部最优点搜寻全局最优点的特性。LINGO 全局算法工具箱的两个基本算法是全局算法 (Global Solver) 和多起点算法 (Multistart Solver)。全局算法使用范围限制和在分支定界体系中的范围缩小技术，把一个非凸的模型转换成一系列比较小的、凸的模型。这种分解-解决 (Divide-and-Conquer) 策略保证最终向全局最优点收敛。另一方面，多起点算法使用试探法从不同的初始点多次重启 NLP 算法。不同的出发点达到不同的局部最优点是很正常的。因此，如果从足够多的点重新开始，就能得到最好的局部解，也就有更大的可能得到真正的全局最优解。

(1) Use Global Solver

如果 Use Global Solver 复选框被选中，当求解一个非线性模型时，LINGO 将调用全局算法。很多非线性模型都是非凸的和 (或) 不光滑的，局部搜索程序的非线性算法 (LINGO 默认的非线

性算法）不适合求解这些模型。一般情况下，这些算法将会集中于局部的次最优点，这个点可能距离真正的全局最优点很远。全局算法使用范围限制的方法（如间隔分析和凸分析）以及在分支定界体系中的范围缩小技术（如线性规划和约束增殖）克服这个弱点，以找到非凸模型的全局解。

下面的例子指出了全局算法的用处。假设一个简单、高度非线性的模型如下：

```
MODEL:
  MIN = X * @COS(3.1416 * X);
  @BND(0,X,6);
END
```

该模型目标函数的图形如图 5-66 所示。

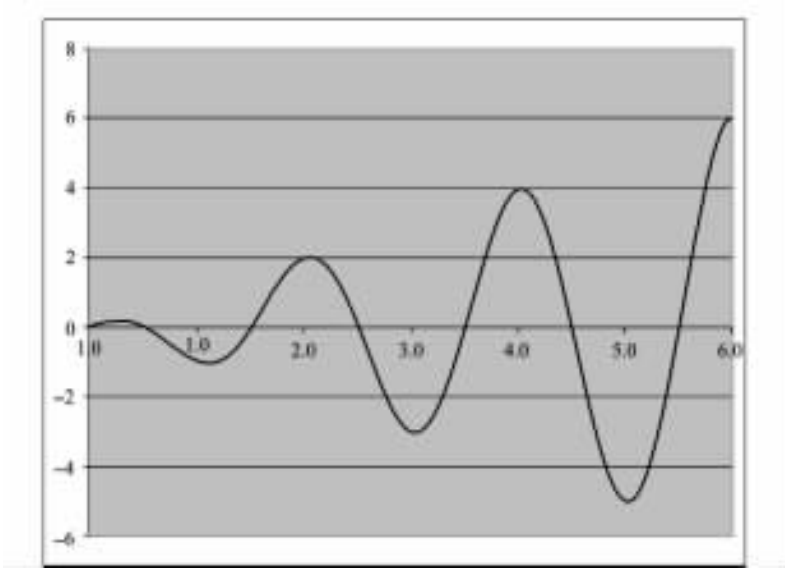



图 5-66　目标函数图形

从图中可以看出，该模型的目标函数在可行的范围内有 3 个局部最小点，并且最右侧的局部点也是全局最优点。如果使用默认的非线性算法，LINGO 将得出如下局部最优解：

```
Local optimal solution found at step:                 11
Objective value:                              -1.046719
     Variable           Value        Reduced Cost
          X        1.090405       0.1181082E-07
        Row    Slack or Surplus      Dual Price
          1          -1.046719       -1.000000
```

如果选择 Use Global Solver 复选框激活全局算法，可以获得如下全局最优解：

```
Global optimal solution found at step:                 35
 Objective value:                              -5.010083
      Variable           Value        Reduced Cost
```

| | | |
|---|---|---|
| X | 5.020143 | -0.7076917E-08 |
| Row | Slack or Surplus | Dual Price |
| 1 | -5.010083 | -1.000000 |

但是,使用全局算法有一个缺点,即比默认的非线性算法的运行速度慢。因此,最好的选择是尽量建立光滑的、凸的非线性模型。默认状态下,全局算法是禁用的。

⑵ Variable Upper Bound

Variable Upper Bound 框用于设置全局算法运行时默认的变量边界。如果该参数设为 d,那么变量的赋值限制在 [- d, d] 之内。在 Value 域内设置该参数要尽可能紧凑,以防止全局算法进入无意义的区间而浪费求解时间。Value 域的默认值是 1e + 010。Application 列表框提供了 None、All 和 Selected 三个选项。选择 None,则完全取消变量的界限,一般不推荐此选项;选择 All,则将使边界应用于所有的变量;选择 Selected,则使全局算法在找到第一个局部解后使用边界,并且边界只应用于不中止于初始局部解的变量。LINGO 的默认设置为 Selected。

⑶ Tolerances

Tolerances 框包含两个被全局算法应用的容差值 Optimality 和 Delta。Optimality 容差值用于指定一个新的解必须在现行解目标值的基础上改变多少才能够成为新的现行解,默认值为 1e-006;Delta 用于指定附加的约束 (作为全局算法凸化过程的一部分) 必须满足的接近程度,默认值为 1e-007。

⑷ Strategies

Strategies 框用于控制全局算法应用的三个策略,即 Branching、Box Selection 和 Reformulation。

Branching 策略包含用于变量第一次分支时用到的六个选项,即 Absolute Width、Local Width、Global Width、Global Distance、Absolute Violation 和 Relative Violation。Branching 框的默认设置为 Relative Violation 选项。

Box Selection 选项用于指定在所有全局算法分支定界树的活动节点中进行选择时采取的策略,包括 Depth First 和 Worst Bound 两个选项,默认的是 Worst Bound 选项。

Reformulation 选项用于设置全局算法执行代数重生成的水平。代数重生成对于构造围绕非线性和非凸方程的紧凸子区域至关重要。可以使用的设置包括 None、Low、Medium 和 High,默认设置是 High。

⑸ Multistart Solver

Global Solvers 页中的 Multistart Solver 框中的 Attempts 选项用于确定 Multistart 算法在试图找到不断改善的局部最优解时重新启动标准 NLP 算法的次数。系统默认选项为 Solver Decides,此时至少启动 5 次以求解小的 NLP 模型,对更大的模型 Multistart 将失活。将 Multistart 值设定为 1, NLP 算法仅仅调用一次,并有效地使 Multistart 失活。将 Multistart 值设定为大于 1 的任何值,NLP 算法将重新启动这么多次以求解所有 NLP 模型。通常,把 Multistarts 值设置为 5 左右对于大多数模型已经足够。高度非线性的模型可能需要一个更大的设定值。

注意:Multistart 算法将急剧增加运行时间,尤其是选择一个大的重复次数时更是如此。因此,应该避免在凸模型中使用不必要的 Multistart。因为这将在没有任何改善的情况下,模型仅沿一条通道向全局最优解收敛。图 5-66 所示的例子同样能够说明 Multistart 算法的用途。

### 5.3.4 Window 菜单

Window 菜单包含一般的管理已打开的窗口命令，内容如图 5-67 所示。

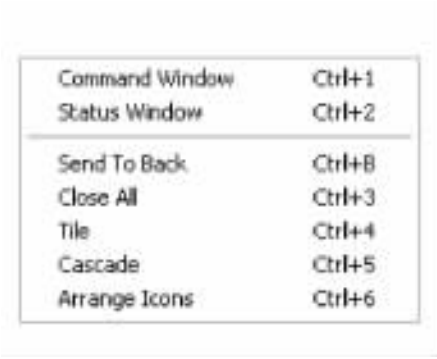



图 5-67 Window 菜单


图 5-68 命令窗口

#### 5.3.4.1 Window| Command Window

Window| Command Window 命令（快捷键 Ctrl + 1）用于打开一个 LINGO 的命令窗口。选择这个命令，图 5-68 所示的窗口就会出现在屏幕上。

这时，可以在窗口的左上角冒号之后输入任何有效的 LINGO 脚本命令。在下面的例子中，用 MODEL 命令输入一个小的模型，并用 LOOK ALL 命令显示所有表达式，然后用 GO 命令求解该模型。含有这些命令脚本的窗口如图 5-69 所示。

一般情况下，最好使用下拉菜单和工具栏执行 LINGO 操作。命令窗口界面主要是为用户交互式地测试命令脚本而设计的。

#### 5.3.4.2 Window| Status Window

如前所述，当调用 LINGO| Solve 命令时，会出现图 1-5 所示窗口。通过此窗口可以监视算法的进程。当状态窗口关闭后，可以通过 Window| Status Window 命令（快捷键 Ctrl + 2）再次将其打开。

#### 5.3.4.3 Window| Send To Back

Window| Send To Back 命令（快捷键 Ctrl + B）用于将活动窗口放至其他所有窗口之后。该命令在模型与结果窗口之间切换时很有用。

#### 5.3.4.4 Window| Close All

Window| Close All 命令（快捷键 Ctrl + 3）用于关闭所有打开的窗口。如果关闭前对一个模型窗口作了改动而没有保存，则系统会提示保存该模型。

#### 5.3.4.5 Window| Tile

Window| Tile 命令（快捷键 Ctrl + 4）用于将所有打开的窗口平铺在屏幕上。每个窗口都被调整了大小，所有的窗口都能显示在屏幕上并且大小基本相等。当运行 Window| Tile 命令时，

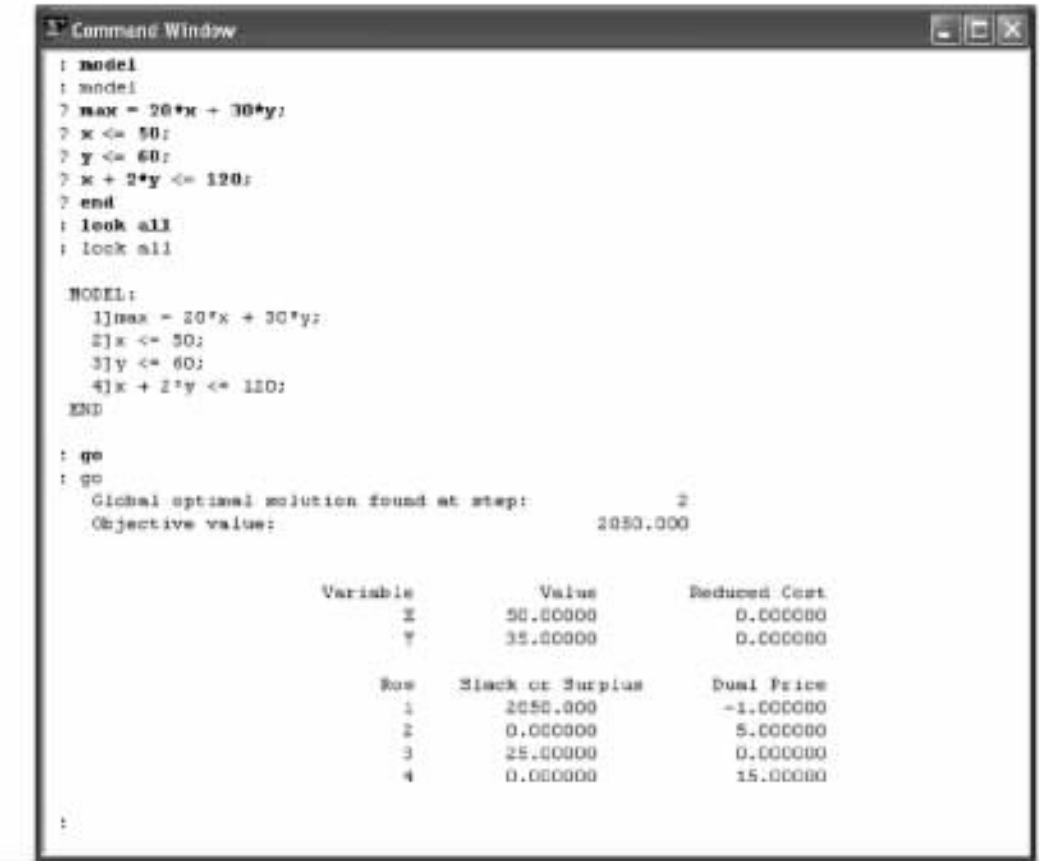



图 5-69　命令脚本举例

会出现如图 5-70 所示对话框。

在该对话框中可以选择将窗口水平 (Horizontal) 或者垂直 (Vertical) 平铺。当水平 (垂直) 平铺时，LINGO 将使每个窗口的水平 (垂直) 尺寸达到最大值。如果打开的窗口多于三个，选择水平或垂直平铺将没有什么区别。

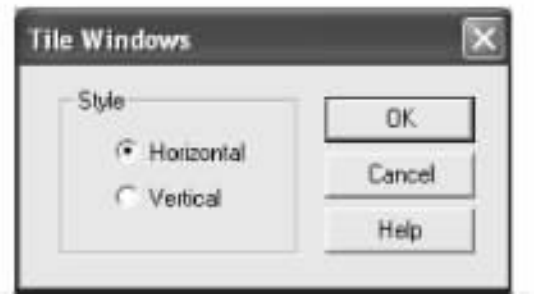



图 5-70　Window| Tile 对话框

5.3.4.6　Window| Cascade

Window| Cascade 命令 (快捷键 Ctrl + 5) 用于将打开的所有窗口层叠式地排在主窗口的左上角，且当前活动窗口在最上面。

5.3.4.7　Window| Arrange Icons

如果将任何一个打开的窗口最小化，它将以图标形式显示在屏幕上，使用 Window| Arrange Icons 命令 (快捷键 Ctrl + 6) 可以将所有的图标排列在窗口框的左下角。


图 5-71　Help 菜单

### 5.3.5　Help 菜单

Help 菜单如图 5-71 所示。该菜单包含了 LINGO 的帮助系统、版权公告和版本特性等。

#### 5.3.5.1　Help | Help Topics

运行 Help Topics 命令 (快捷键 Ctrl + 6) 后显示的对话框如图 5-72 所示。


图 5-72　Help Topic 对话框

选择 Contents 页显示帮助系统的内容目录，通过双击可以选择感兴趣的任何帮助信息。选择 Index 页显示帮助系统的索引目录，通过双击可以浏览选择的项。选择 Find 页可以查找帮助系统的详细项目。

#### 5.3.5.2　Help | Register

使用 Help | Register 命令可以在线注册 LINGO 版本。当运行 Register 命令时，需要连接因特网，并且会出现图 5-73 所示对话框。

输入个人信息并点击 Register 按钮，个人信息通过因特网直接发送到 LINDO Systems 公司。注册成功后，在屏幕上出现图 5-74 所示对话框。点击 OK 按钮回到 LINGO 的工作环境。


图 5-73　Register 对话框

5.3.5.3　Help | AutoUpdate


图 5-74　注册完成对话框

打开 Help | AutoUpdate 命令后，每次使用 LINGO 软件时都自动检测在 LINDO Systems 网站是否有 LINGO 的最新版本可以下载。同样，要想使用这个命令，必须连接到互联网上。

当使用 AutoUpdate 命令或者用 AutoUpdate 命令激活一个 LINGO 版本时，LINGO 将会搜索公司网站察看是否有更近版本的 LINGO 软件可以下载。如果目前使用的是最新版本或 AutoUpdate 处于休眠模式，那么将会回到 LINGO 环境。如果使用的是一个旧版本软件，将会出现图 5-75 所示对话框。

这时，可以在文本框中输入天数并点击 Remind me in 按钮使 AutoUpdate 处于休眠模式。这样在 Help 菜单中，该命令旁边会出现一个复选标记，表明其处于激活状态。同时，也可以点击 Yes 按钮从网站下载软件的最新版本。LINGO 将下载新的安装文件，并且出现图 5-76 所示窗口显示下载进度。

下载完毕后，将出现图 5-77 所示对话框。

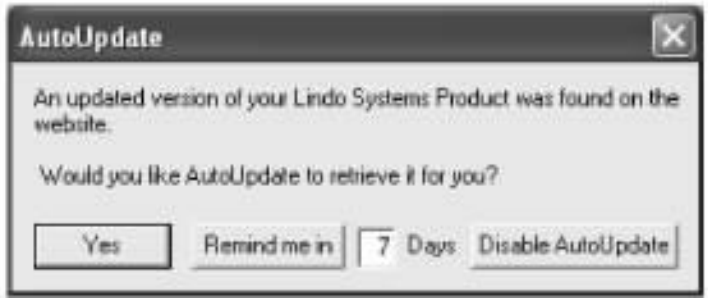



图 5-75　AutoUpdate 对话框

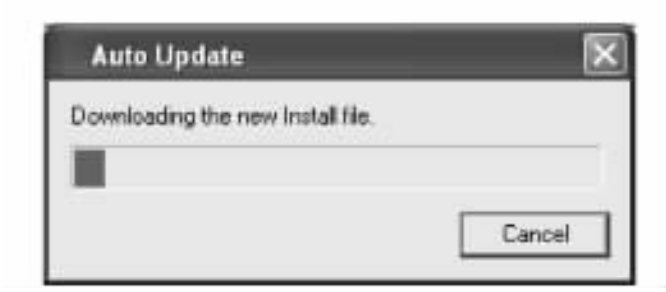



图 5-76　下载进度窗口

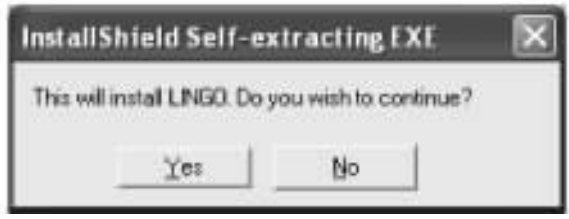



图 5-77　下载完毕对话框

点击 Yes 按钮，打开 InstallShield Wizard，通过安装完成 LINGO 的更新。为了方便起见，在安装过程中注册密码将自动加载到新版本中。

如果在 AutoUpdate 对话框中点击 Disable AutoUpdate 按钮，则 AutoUpdate 的所有特征失活。在默认状态下，AutoUpdate 处于失活状态。

5.3.5.4　Help | About LINGO

当运行 About LINGO 命令时，会出现图 5-78 所示对话框。

在 About LINGO 对话框的第一部分列出了 LINGO 副本的级别及版本信息。第二部分列出了联系 LINDO Systems 公司的方法。第三部分标题为 Limits for this Installation，列出了相应版本的各种能力限制，而且指出了分配给模型生成器的内存数。对于各种版本目前的规定如表 5-6 所示。第四部分标题为 License Expiration，列出了注册期限。如果许可证没有产品有效期，这部分将显示 Perpetual (永久)。License Usage 标签中列出了注册版本是用于商业还是教育。教育的许可证仅限教育机构用于教育和研究目的，商业的许可证没用特定的使用限制。Number

of Licenses 框列出了可以使用该 LINGO 副本的用户数。最后，Additional License Information 框列出了注册版本的相关附加信息。多数情况下，用户的 LINGO 序列号可以在这里找到，也可以通过这个区域的滚动条找到其他一些附加信息，包括版本功能（如障碍算法和非线性算法）。

表 5-6　LINGO 版本信息

| 版本 | 总变量数 | 整型变量数 | 非线性变量数 | 约束条件数 |
|---|---|---|---|---|
| Demo/Web | 300 | 30 | 30 | 150 |
| SolverSuite | 500 | 50 | 50 | 250 |
| Super | 2 000 | 200 | 200 | 1 000 |
| Hyper | 8 000 | 800 | 800 | 4 000 |
| Industrial | 320 003 200 | 3 200 | 16 000 | |
| ExtendedUnlimited | Unlimited | Unlimited | Unlimited | |

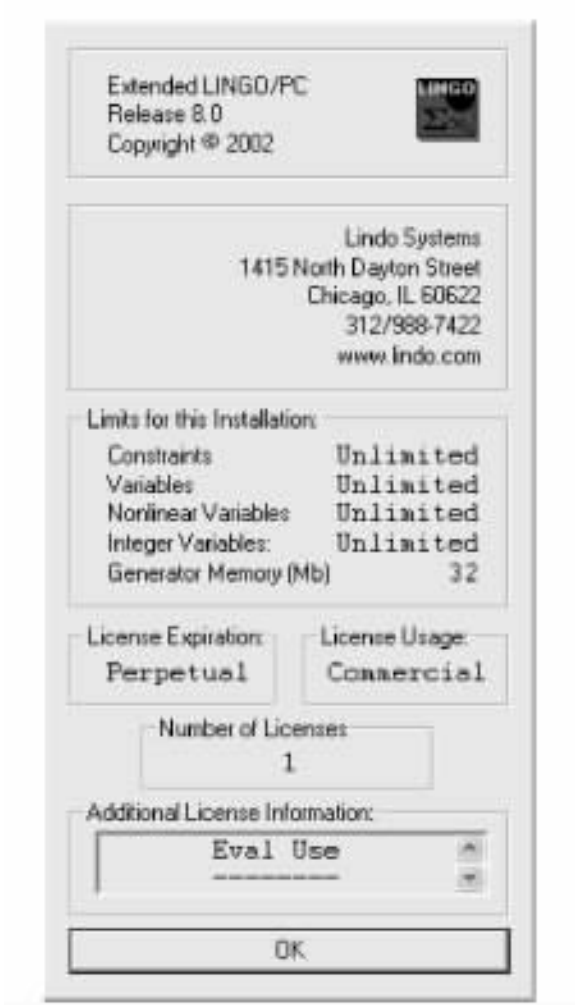



图 5-78　About LINGO 对话框

5.3.5.5 Help | Pointer 

**按下这个按钮可把光标转化成** Help **模式。光标变成** Help **模式后，可以选择菜单命令或者工具栏按钮，**LINGO **将显示与其相应的帮助信息。**

# 第 6 章　命令行命令

在本章中将讨论 LINGO 的命令行命令。在非 Windows 的操作系统中，大多需要使用 LINGO 的命令行命令。而在 Windows 版本下，命令行命令可以通过命令窗口（Command Window）输入，也可以通过建立命令脚本实现。命令脚本可以在 LINGO 启动时自动运行或者在用户需要的时候运行，从而完成一些特定的功能。

## 6.1　命令的简要介绍

首先根据命令行命令的功能将其进行分类。每个类别的命令及功能如表 6-1 至表 6-9 所示。

表 6-1　信息命令

| 命令 | 功能 |
|---|---|
| CAT | 列举可以利用的命令的类别 |
| COM | 按类别列举可以利用的命令 |
| HELP | 提供命令的简要帮助 |
| MEM | 显示工作内存使用状况的统计信息 |

表 6-2　输入命令

| 命令 | 功能 |
|---|---|
| FRMPS | 重新读取一个自由 MPS 格式的模型 |
| MODEL | 开始输入一个新的模型 |
| RMPS | 重新读取一个固定 MPS 格式的模型 |
| TAKE | 运行外部文件上的命令脚本 |

表 6-3　显示命令

| 命令 | 功能 |
|---|---|
| GEN | 生成代数形式的模型 |
| GENL | 生成符合 LINDO 格式的代数形式的模型 |
| HIDE | 运用口令保护当前模型 |
| LOOK | 显示当前模型 |
| PICTURE | 显示模型非零结构的图形 |
| STATS | 对产生的模型特征做简要的统计 |

表 6-4　文件输出命令

| 命令 | 功能 |
|---|---|
| DIVERT | 打开一个接收输出信息的文件 |
| RVRT | 关闭一个由 DIVERT 命令打开的输出文件 |
| SAVE | 将当前模型保存到磁盘上 |
| SMPI | 输出 MPI 格式的模型 |
| SMPS | 发送目前模型的副本到 MPS 格式的文件中 |
| HIDE | 将模型的文体设置为隐藏或非隐藏状态 |

表 6-5　求解命令

| 命令 | 功能 |
|---|---|
| DEBUG | 在不可行和无界的模型中找出表达式错误 |
| GO | 求解当前模型 |
| NONZ | 仅产生一个非零的结果报告 |
| RANGE | 产生一个灵敏度分析报告 |
| SOLU | 生成一个结果报告 |

表 6-6　编辑命令

| 命令 | 功能 |
|---|---|
| ALTER | 编辑当前的模型 |
| DELETE | 删除当前模型中选中的行 |
| EXTEND | 在当前模型的末尾增加行 |

表 6-7　交互参数命令

| 命令 | 功能 |
|---|---|
| PAGE | 设置页长或者屏幕显示的长度 |
| PAUSE | 暂停键盘输入 |
| TERSE | 转为概要输出模式 |
| VERBOSE | 转为详细输出模式 |
| WIDTH | 设置终端显示和输入的宽度 |

表 6-8　设置命令

| 命令 | 功能 |
|---|---|
| DBUID | 设置经 @ODBC 访问数据库的用户 ID |
| DBPWD | 设置经 @ODBC 访问数据库的密码 |
| FREEZE | 保存当前设置 |
| SET | 重新设定 LINGO 若干默认值和容差值 |

表 6-9　其他命令

| 命令 | 功能 |
|---|---|
| ! | 插入注释 |
| NEWPW | 输入新的密码以更新许可信息 |
| QUIT | 退出 LINGO |
| TIME | 显示求解开始后至今所用的时间 |

## 6.2　命令的详细介绍

这一部分将对 LINGO 的每一个命令行命令进行详细介绍。

### 6.2.1　信息命令

#### 6.2.1.1　CAT

CAT 命令用于显示 LINGO 中可以利用的九类命令行命令。输入 CAT 命令后，可以按照提示输入与各类别相对应的数字，LINGO 将显示该类别下的所有命令。要退出 CAT 命令，输入一个空白行即可。

#### 6.2.1.2　COM

COM 命令用于按类别列出 LINGO 所有的命令行命令。

#### 6.2.1.3　HELP

HELP 命令要与 LINGO 的另一个命令行命令联合使用，用于给出该指定命令的简要信息。没有参数的 HELP 命令，将返回 LINGO 的版本信息以及与该版本相对应的最大变量数目和最大约束条件数目等信息。

#### 6.2.1.4　MEM

MEM 命令用于显示模型生成器工作内存使用状况的统计信息。下面是使用 MEM 命令的一个例子。

```
:MEM
Total generator memory          5242880
Peak generator memory usage       12048
Current generator memory usage     1312
Total handles                        96
Peak handle usage                     9
Current handle usage                  5
Total bytes moved                  1552
```

```
Total blocks moved                    6
Total heap compacts                   0
Fragmentation ratio               0.002
:
```

其中，Total generator memory（总内存数）表示 LINGO 为生成一个模型已经分配的内存数，可以用 SET 命令控制该值的大小；Peak generator memory usage（内存使用量峰值）表示目前处理过程中内存的最大使用量；Current generator memory usage（当前内存使用量）表示当前处理过程使用的内存数量；Total handles（总句柄数）是 LINGO 可分配内存块的最大数量；Peak handle usage（句柄使用量峰值）表示 LINGO 在某一处理过程中内存块的最大使用量；Current handle usage（当前句柄使用量）表示此刻正在使用的内存块的数量；Total bytes moved（字节移动总数）给出了到目前为止模型生成器为了重新分配内存而必须移动的内存字节数；Total blocks moved（内存块移动总数）给出了重新分配内存而移动的内存块总数；Total heap compacts（内存堆压缩倍数）给出模型生成器为了给不断增长的内存需求腾出空间而压缩内存堆的倍数，如果内存堆压缩倍数过大，应该使用 SET 命令分配更多的工作内存；Fragmentation ratio（碎片比例）是一个统计值，用于衡量内存堆的破碎程度。1 代表高的破碎程度，0 代表没有破碎现象。

## 6.2.2 输入命令

### 6.2.2.1 FRMPS 和 RMPS

FRMPS 和 RMPS 命令用来读取 MPS 格式的模型。MPS 文件格式是由 IBM 公司提出的一种行业标准格式，它使得模型可以从一种算法或平台中转移到另外的算法或平台。

目前，FRMPS 和 RMPS 命令还不支持二次及以上的 MPS 格式文件，因此被这两个命令读取的模型必须是线性模型或者混合整型线性模型。FRMPS 命令用于读取自由格式的 MPS 文件，而 RMPS 命令用于读取固定格式的 MPS 文件。

当读取 MPS 文件时，LINGO 将把 MPS 格式转换成等价的 LINGO 格式。考虑下面的简单模型：

```
ObjRow) Maximize 20X + 30Y
Subject To:
  Row1) X < 50
  Row2) Y < 60
  Row3) X + 2Y < 120
```

与上述模型等价的 MPS 文件如下：

```
NAME          SAMPLE
OBJSENSE
  MAX
ROWS
  N OBJROW
  L ROW1
```

```
  L ROW2
  L ROW3
COLUMNS
    X         ROW3          1.0000000
    X         OBJROW       20.0000000
    X         ROW1          1.0000000
    Y         OBJROW       30.0000000
    Y         ROW2          1.0000000
    Y         ROW3          2.0000000
RHS
    RHS       ROW1         50.0000000
    RHS       ROW2         60.0000000
    RHS       ROW3        120.0000000
ENDATA
```

可见，MPS 格式对于存储模型来说并不是紧凑的格式。

在下面的例子中，用 LINGO 读取 MPS 文件，并用 LOOK 命令显示这个模型，可以看到模型是如何自动从 MPS 格式转换到 LINGO 格式的。例子如下：

```
:rmps c:\sample.mps
:look all
  1] TITLE  SAMPLE;
  2] [OBJROW] MAX = 20 * X + 30 * Y;
  3] [ROW1]  X < = 50;
  4] [ROW2]  Y < = 60;
  5] [ROW3]  X + 2 * Y < = 120;
```

如果想以 MPS 格式而非 LINGO 格式保存这个文件，可以用 SMPS 命令。

当涉及约束条件和变量的名称限制时，MPS 格式比 LINGO 格式的限制少。MPS 中的对象名称允许内部含有空格和其他额外的字符。为了弥补这个差别，在读取 MPS 文件时，LINGO 会尽量修复对象名称，以便符合 LINGO 的语法。实际操作中，LINGO 用下画线来代替名称中不符合语法的字符。大多数情况下，这种方法是有效的。然而，这样做有可能会出现名称冲突，即两个或更多的名称在修改后变为同一名称。例如，变量 X.1 和 X%1 都会被替代成同一个 LINGO 的名称 X_1。这种情况会完全改变模型的结构而出现如下错误：

```
[Error Code: 179]
The MPS reader had to patch names to make them compatible:
  var names patched:           1
  row names patched:           0
Name collisions may have occurred.
```

该信息显示了在修改名称以便符合 LINGO 语法时，被修改的变量和行的个数。

LINGO 还提供了一种特定的方法解决名称冲突的问题，该方法包括在 MPS I/O 中，遇到名称问题时采用 RC 格式。RC 格式包括将模型中的行（约束条件）重新命名为 Rn，其中 n 是行的

索引值。同样,将每列(变量)重新命名为 Cn。此外,还将目标函数行重新命名为 ROBJ。为了把 MPS 格式的名称转化成 RC 格式,使用如下语句:

```
:SET RCMPSN 1
```

这将使得 LINGO 在读取和保存所有的 MPS 文件时都使用 RC 命名规则。若要取消 RC 命名规则,则输入以下语句:

```
:SET RCMPSN 0
```

举个例子,再次读取上文中曾读取过的 MPS 格式的模型。但是,这一次打开 RC 命名规则:

```
:set rcmpsn 1
Parameter        Old Value     New Value
  RCMPSN             0             1
:rmps c:\sample.mps
:look all
  1] TITLE  SAMPLE;
  2] [ROBJ] MAX = 20 * C1 + 30 * C2;
  3] [R1]  C1 < = 50;
  4] [R2]  C2 < = 60;
  5] [R3]  C1 + 2 * C2 < = 120;
```

可见,变量名称应用 RC 格式后保证了不会发生名称冲突。

另一个潜在的冲突是 MPS 允许约束条件的名称和变量的名称相同,LINGO 却不允许这样。当求解模型时,若出现错误信息 28(行名称无效)或者错误信息 37(名称已被使用),同样可以使用 RC 命名规则来避免这种冲突。

#### 6.2.2.2 MODEL

MODEL 命令用于在 LINGO 中开始输入一个新的模型。LINGO 为该模型的每一个输入行给出一个提示符"?"。模型输入完毕后,在单独一行中输入 END 表示结束,然后 LINGO 回到正常的命令模式(显示一个冒号":")。在下面的例子中,将用 MODEL 命令输入一个小的模型,然后用 LOOK 命令显示其内容并用 GO 命令对其进行求解。模型如下:

```
:MODEL
?! How many years does it take
?to double an investment growing
?10 % per year?;
?1.1 ^ YEARS = 2;
?END
:LOOK ALL
  1]! How many years does it take
  2]to double an investment growing
```

```
  3]10 % per year ?;
  4]1.1 ^ YEARS = 2 ;
:GO
Feasible solution found at step:    0
         Variable           Value
            YEARS        7.272541

              Row    Slack or Surplus
                1            0.000000
```

6.2.2.3 TAKE

TAKE 命令有两个主要作用：一是读取用 SAVE 命令保存在磁盘中的模型文件；二是执行包含在外部文件中的命令脚本。TAKE 命令的语法如下：

```
TAKE [filename]
```

如果忽略了文件名称，LINGO 将给出提示。

假设使用 SAVE 命令把一个模型保存到 C：h LINGOMOD h MYMODEL. LNG 下，则可以用下面的命令将它读取到 LINGO 中：

```
:TAKE C:\ LINGOMOD \ MYMODEL.LNG
```

用 TAKE 命令还可以执行 LINGO 中的一个命令脚本。脚本文件是一个包含一系列 LINGO 命令的简单文本文件。假设在一个编辑器中建立了如下的命令脚本，保存为文本文件，路径是 D：h LNG h MYSCRIPT. LTF。该文件内容如下：

```
MODEL:
! For a given probability P, this
model returns the value X such
that the probability that a unit
normal random variable is less
than or equal to X is P;
!Here is the probability;
  P = .95;
  !Solve for X;
  P = @PSN(X);
END
!Terse output mode;
TERSE
!Solve the model;
GO
!Report X;
SOLU X
```

用 TAKE 命令运行该脚本语句如下：

```
:TAKE D:\LNG\MYSCRIPT.LTF
Feasible solution found at step:     0
          Variable           Value
             X            1.644854
:
```

## 6.2.3 显示命令

### 6.2.3.1 GEN

GEN 命令用于编译当前的模型，以产生一个原始的、展开的代数形式的模型，并显示在屏幕上。

当调试一个问题时，使用 GEN 命令产生的输出很有帮助。扩展开的表达式对于发现模型逻辑上的错误和不合理的设想很有益处。为了说明 GEN 命令的使用，考察下面的运输模型：

```
MODEL:
SETS:
  SUPPLY/WH1,WH2,WH3/:CAP;
  DEST/C1,C2,C3,C4/:DEM;
  LINKS(SUPPLY,DEST):COST,VOL;
ENDSETS
MIN = @SUM(LINKS:COST*VOL);
@FOR(DEST(J):[INTO]
@SUM(SUPPLY(I):VOL(I,J)) > =
  DEM(J)
);
@FOR(SUPPLY(I):[FROM]
@SUM(DEST(J):VOL(I,J)) < =
  CAP(I)
);
DATA:
  CAP = 30,25,21;
  DEM = 15,17,22,12;
  COST = 6,2,6,7,
        4,9,5,3,
        8,8,1,5;
ENDDATA
END
```

用 GEN 命令运行这个模型，将会得到如下的结果：

```
MIN       5 VOL(WH3,C4) + VOL(WH3,C3) +
```

```
  8 VOL(WH3,C2) + 8 VOL(WH3,C1) +
  3 VOL(WH2,C4) + 5 VOL(WH2,C3) +
  9 VOL(WH2,C2) + 4 VOL(WH2,C1) +
  7 VOL(WH1,C4) + 6 VOL(WH1,C3) +
  2 VOL(WH1,C2) + 6 VOL(WH1,C1)
SUBJECT TO
INTO(C1)]  VOL(WH3,C1) + VOL(WH2,C1) +
  VOL(WH1,C1) > = 15
INTO(C2)]  VOL(WH3,C2) + VOL(WH2,C2) +
  VOL(WH1,C2) > = 17
INTO(C3)]  VOL(WH3,C3) + VOL(WH2,C3) +
  VOL(WH1,C3) > = 22
INTO(C4)]  VOL(WH3,C4) + VOL(WH2,C4) +
  VOL(WH1,C4) > = 12
FROM(WH1)]  VOL(WH1,C4) + VOL(WH1,C3) +
  VOL(WH1,C2) + VOL(WH1,C1) < = 30
FROM(WH2)]  VOL(WH2,C4) + VOL(WH2,C3) +
  VOL(WH2,C2) + VOL(WH2,C1) < = 25
FROM(WH3)]  VOL(WH3,C4) + VOL(WH3,C3) +
  VOL(WH3,C2) + VOL(WH3,C1) < = 21
END
```

**正如上面所看到的,在生成的模型里,所有最优化问题中的变量和行都被明确地显示出来了。**

**如果希望将** GEN **命令产生的结果放到一个文件里,可以在使用** GEN **命令之前,用** DIVERT **命令打开一个输出文件。**

#### 6.2.3.2 GENL

GENL **命令用于生成与** LINDO **相容的模型,除了这点,**GENL **和** GEN **命令基本相同。具体地讲,为了符合** LINDO **命名规则中对约束和变量名称** 8 **个字符的限制,**GENL **会将其多余部分去掉。**

**例如,用** GENL **命令来执行上面** GEN **命令中的运输模型,**LINGO **将生成如下输出:**

```
MIN   5 VOLWH3C4 + VOLWH3C3 + 8 VOLWH3C2 +
  8 VOLWH3C1 + 3 VOLWH2C4 + 5 VOLWH2C3 +
  9 VOLWH2C2 + 4 VOLWH2C1 + 7 VOLWH1C4 +
  6 VOLWH1C3 + 2 VOLWH1C2 + 6 VOLWH1C1
SUBJECT TO
INTO(C1)  VOLWH3C1 + VOLWH2C1 + VOLWH1C1
  > = 15
INTO(C2)  VOLWH3C2 + VOLWH2C2 + VOLWH1C2
  > = 17
```

```
INTO (C3)  VOLWH3C3 + VOLWH2C3 + VOLWH1C3
  > = 22
INTO (C4)  VOLWH3C4 + VOLWH2C4 + VOLWH1C4
  > = 12
FROM (WH1)  VOLWH1C4 + VOLWH1C3 + VOLWH1C2
  + VOLWH1C1 < = 30
FROM (WH2)  VOLWH2C4 + VOLWH2C3 + VOLWH2C2
  + VOLWH2C1 < = 25
FROM (WH3)  VOLWH3C4 + VOLWH3C3 + VOLWH3C2
  + VOLWH3C1 < = 21
END
```

**可见,为了符合 LINDO 命名规则中 8 个字符长度的限制,名称中的圆括号和逗号都被删除了。同样,为了将扩展的运输模型输送到文件 TRANSPRT.LTX 中,要使用下面的语句:**

```
:DIVERT TRANSPRT.LTX
:GENL
:RVRT
```

**还可以使用 LINGO 的 TAKE 命令将 TRANSPRT.LTX 文件加载到 LINGO 中。**

**注意:当将 LINGO 模型移植到 LINDO 下运行时,最好使用较短的变量名。由于 LINDO 结果中的变量名称有 8 个字符长度的限制,所以,如果在上面模型中使用 VOLM 而不是 VOL,则使用 GENL 命令后,VOLM(WH1,C1) 和 VOLM(WH1,C2) 将都被替换成 VOLMWH1C,这样将出现变量名称的冲突。**

#### 6.2.3.3 LOOK

**LOOK 命令用于显示当前模型的全部或者一部分。LOOK 命令的语法是:**

```
LOOK row_index | beg_row_index end_row_index | ALL
```

**这样就可以用行的索引号查看一部分或者用 ALL 查看整个模型。**

**在下面的例子中,用 LOOK 命令的几种形式来观察模型:**

```
:LOOK ALL
  1] !For a given probability P, this
  2] model returns the value X such
  3] that the probability that a unit
  4] normal random variable is less
  5] than or equal to X is P;
  6]
  7] !Here is the probability;
  8] P = .95;
  9]
 10] !Solve for X;
```

```
11] P = @PSN(X);
12]
:LOOK 8
 8] P = .95;
:LOOK 10 11
10] !Solve for X;
11] P = @PSN(X);
```

### 6.2.3.4 PICTURE

**PICTURE 命令用于以矩阵形式显示模型。对于规模不大的模型来说，PICTURE 命令在获得模型的可视化表达方面很有用，并且有利于寻找表达式中的错误。**

**首先，使用表 6-10 所示的字母代码（Letter Code）来代表 PICTURE 命令的输出中的线性参数。**

**表 6-10　字母代码与系数范围**

| 字母代码 | 系数范围 | 字母代码 | 系数范围 |
|---|---|---|---|
| Z | (0.000 000,0.000 001] | A | (1,10] |
| Y | (0.000 001,0.000 01] | B | (10,100] |
| X | (0.000 01,0.000 1] | C | (100,1 000] |
| W | (0.000 1,0.001] | D | (1 000,10 000] |
| V | (0.001,0.01] | E | (10 000,100 000] |
| U | (0.01,0.1] | F | (100 000,1 000 000] |
| T | (0.1,1) | G | > 1 000 000 |

**值得注意的是，单个数字的整数将直接显示，而不显示为字母。如果一行中的某一个变量是非线性的，则 PICTURE 命令将用问号代表其参数。**

**在下面的例子中，读取运输模型并用 PICTURE 命令观察模型的逻辑结构：**

```
:take \lingo8\samples\tran.lg4
:pic
          V V V V V V V V V V V V
          O O O O O O O O O O O O
          L L L L L L L L L L L L
          U U U U U U U U U U U U
          M M M M M M M M M M M M
          E E E E E E E E E E E E
          ( ( ( ( ( ( ( ( ( ( ( (
          W W W W W W W W W W W W
          H H H H H H H H H H H H
          1 1 1 1 2 2 2 2 3 3 3 3
```

```
                ' ' ' ' ' ' ' ' ' ' ' ' '
                C C C C C C C C C C C C C
                1 2 3 4 1 2 3 4 1 2 3 4
                ) ) ) ) ) ) ) ) ) ) ) ) )
        OBJ:    6 2 6 7 4 9 5 3 8 8 1 5  MIN
   DEM(C1):     1     ' 1   '   1 '       > B
   DEM(C2):     ' 1 ' ' ' 1 ' ' 1     '   > B
   DEM(C3):     '   1 '     1     ' 1     > B
   DEM(C4):     '     1     ' 1   '   1   > B
   SUP(WH1):    1 1 ' 1 1 '   ' ' ' '     < B
   SUP(WH2):    '     ' 1 1 1 1   '       < B
   SUP(WH3):    '     '     '   1 1 1 1   < B
```

在模型中,右边的所有值在范围 [20,30] 之间。因此,用 B 来代替所有的这些值。在矩阵左边,从上到下依次是每行的名称,变量名称显示在最上部。目标函数行和每个约束条件的系数也被显示出来。其中的空格代表 0,而插入的单引号可以使背景呈现栅格状。

注意:PICTURE 命令最适合应用在较小的模型上,若用在较大的模型上将会使输出冗长。对于较大模型来说,Windows 版本的 LINGO 中的 PICTURE 命令能够将矩阵图片压缩成一屏以便查看。

#### 6.2.3.5 STATS

STATS 命令用于列出模型的概要统计信息,其用法和信息的含义参见 LINGO | Model Statistics 菜单中的介绍。

### 6.2.4 文件输出命令

#### 6.2.4.1 DIVERT 和 RVRT

DIVERT 命令用于打开一个文件,使 LINGO 将之后所有的报告 (例如 SOLUTION、RANGE 和 LOOK 命令生成的报告) 从屏幕发送到该文件。DIVERT 命令在这个指定的文件里用文本格式存放报告。由于 DIVERT 命令产生的文件是文本格式的,所以它可以被其他程序使用。DIVERT 命令的语法如下:

```
DIVERT filename
```

其中,filename 为要创建的文件名。

RVRT 命令与 DIVERT 命令相反,它用于在关闭 DIVERT 命令打开的文件之后将结果重新输出到屏幕上。

在下面的例子中,继续用 MODEL 命令生成一个小模型,用 GO 命令对其求解,然后用 DIVERT 命令创建一个包含问题和求解结果的文件。例子如下:

```
:! Enter a small model
:MODEL
```

```
? MAX = 20 * X + 30 * Y;
? X < = 50;
? Y < = 60;
? X + 2 * Y < = 120;
? END
:! Solve the model
:TERSE
:GO
Global optimal solution found at step:        1
Objective value:                          2050.000
:! Create a DIVERT file with
:! the formulation & solution
:DIVERT MYFILE.TXT   ! Opens the file
:LOOK ALL            ! Sends model to file
:SOLU                ! Sends solution to file
:RVRT                ! Closes DIVERT file
:
```

**打开由** DIVERT **命令创建的** MYFILE.TXT **文件,可以看到下面的表达式和求解结果:**

```
  1]MAX = 20 * X + 30 * Y;
  2]X < = 50;
  3]Y < = 60;
  4]X + 2 * Y < = 120;
Variable           Value        Reduced Cost
     X           50.00000            0.000000
     Y           35.00000            0.000000
   Row    Slack or Surplus      Dual Price
     1           2050.000            1.000000
     2           0.000000            5.000000
     3           25.00000            0.000000
     4           0.000000            15.00000
```

**注意:**

1) **当** DIVERT **命令生效时,在屏幕上很少或者根本看不到输出,这是由于大部分输出已被送到** DIVERT **指定文件中;**

2) **要确保选择的** DIVERT **命令操作的文件名与模型的文件名不同,否则,输出将覆盖模型文件且不能恢复。**

#### 6.2.4.2 SAVE

SAVE **命令用于将当前的模型保存到文件中,语法如下:**

```
SAVE filename;
```

其中，filename 是保存模型的文件名。LINGO 以文本格式保存模型，并且可以用 TAKE 命令把模型读回到 LINGO 中。对模型文件最好采用扩展名 .LNG，这样可以很容易识别它们。

若想用自己的文本编辑器修改文件，一定要确保 LINGO 模型是用文本格式保存的。

在下面的例子中，首先输入一个小模型，然后用 MYMODEL.LNG 文件保存该模型。例子如下：

```
:! Enter a small model
:MODEL
? MAX = 20 * X + 30 * Y;
? X < = 50;
? Y < = 60;
? X + 2 * Y < = 120;
? END
:! Save model to a file
:SAVE MYMODEL.LNG
:
```

如果用文本编辑器打开模型文件 MYMODEL.LNG，将会看到如下的内容：

```
MODEL:
  1] MAX = 20 * X + 30 * Y;
  2] X < = 50;
  3] Y < = 60;
  4] X + 2 * Y < = 120;
END
```

#### 6.2.4.3 SMPI

SMPI 命令用于以一个特殊的格式保存模型，该格式名为 Mathematical Programming Interface (MPI)。MPI 是由 LINDO 公司开发出的一种特殊格式，用于表示所有类别的数学模型——线性的、整数的和非线性的。但是，由于 LINGO 目前不能读取 MPI 格式的文件，所以不宜使用 MPI 格式永久地保存模型，应用 SAVE 命令来保存文件，以备后用。

#### 6.2.4.4 SMPS

SMPS 命令用于为当前模型产生一个内在的代数表达形式，并用 MPS 格式将其保存到一个文件中。MPS 格式是一个表示线性规划的通用格式，具有该格式的文件可以被任何识别 MPS 文件的求解器阅读，包括其他大多数普通的线性规划软件包。

SMPS 命令的语法如下：

```
SMPS filename;
```

其中，filename 是 MPS 格式保存模型的文件名。

在下面的例子中，输入一个小模型，然后用 MPS 文件保存它。例子如下：

```
:! Enter a small model
:MODEL
? MAX = 20 * X + 30 * Y;
? X < = 50;
? Y < = 60;
? X + 2 * Y < = 120;
? END
:! Save model to an MPS file
:SMPS MYMODEL.MPS
:
```

**如果用一个文本编辑器打开该 MPS 文件,将会得到如下结果:**

```
NAME     LINGO GENERATED MPS FILE (MAX)
ROWS
  N 1
  L 2
  L 3
  L 4
COLUMNS
    Y         1          30.0000000
    Y         3           1.0000000
    Y         4           2.0000000
    X         1          20.0000000
    X         2           1.0000000
    X         4           1.0000000
RHS
    RHS       2          50.0000000
    RHS       3          60.0000000
    RHS       4         120.0000000
ENDATA
```

**注意:**

**1) 模型必须是线性的才可以用 SMPS 格式成功输出。如果模型是非线性的,在 MPS 文件中,将把非线性处的系数替换成为一个问号。**

**2) 如前所述,SMPS 将截去变量名中 8 个字符以外的所有字符,所以选择变量名时应注意避免截名后发生冲突。**

**3) MPS 文件格式的用途主要是为了输出模型到其他应用软件或平台上,而且 MPS 格式在性质上是纯粹的数量格式。当将 LINGO 模型转换成 MPS 格式时,所有基于集合的信息都会丢失。因此,应该使用 SAVE 命令而不是 SMPS 命令保存模型的副本。**

#### 6.2.4.5 HIDE

**HIDE 命令用于隐藏模型的文本。如果希望保护模型的创作权,这个命令会很有帮助。**

当输入 HIDE 命令后，将出现一个口令提示，这时可以输入任何 8 个字符作为口令。LINGO 将再次提示输入以确认该口令，并且口令中的字母是区分大小写的。

模型被隐藏后，显示模型文本的命令（如 GEN、GENL、LOOK 和 SMPS 等）将都失去作用，其他的命令（除了 ALTER）仍发挥正常的作用。如果一个模型被隐藏，ALTER 命令仍可以修改模型，但不会反映到屏幕上。

当一个处于隐藏状态的模型被保存后，它的文本就会被加密。这样可以防止用 LINGO 之外的方式查看模型。

只要再次使用 HIDE 命令并输入正确的口令，一个隐藏的模型就可以返回到正常非隐藏的状态。举例说明 HIDE 命令的应用如下：

```
:TAKE D:\LNG\TRAN.LNG   ! Read in a model
:LOOK 4 6               ! Display some rows
  4]   SUPPLY/WH1,WH2,WH3/  :CAP;
  5]   DEST/C1,C2,C3,C4/ :DEM;
  6]   LINKS(SUPPLY,DEST) :COST,VOL;
:HIDE          ! Now hide the model
Password?
TIGER
Please reenter password to verify:
TIGER
Model is now hidden.
:! Model is hidden so LOOK will fail
:LOOK ALL
  [Error Code: 111]
Command not available when model is hidden.
:TERSE
:GO
Global optimal solution found at step:        6
Objective value:                        161.0000
:! And get a solution report
:NONZ VOL
     Variable           Value        Reduced Cost
VOL(WH1,C1)           2.000000       0.000000
VOL(WH1,C2)           17.00000       0.000000
VOL(WH1,C3)           1.000000       0.000000
VOL(WH2,C1)           13.00000       0.000000
VOL(WH2,C4)           12.00000       0.000000
VOL(WH3,C3)           21.00000       0.000000
:! Now, unhide the model
:HIDE
Password?
```

```
TIGER
:! Once again, we can view the model
:LOOK 4 6
  4]   SUPPLY/WH1,WH2,WH3/   :CAP;
  5]   DEST/C1,C2,C3,C4/ :DEM;
  6]   LINKS(SUPPLY,DEST) :COST,VOL;
```

## 6.2.5 求解命令

### 6.2.5.1 DEBUG

理想状态下,所有的模型都有最优解,而事实上并非如此。在很多情况下,都可能遇到不可行或者无界的模型。特别是在项目开发阶段,当模型中有文字错误时,这种情况尤其显著。在大型模型中查找错误是一件很繁琐的事情,而 DEBUG 命令可以减少查找错误的困难。使用该命令后,原始模型中有一部分被当作问题源隔离出来,这就有利于集中精力在部分模型中寻找表达式或者输入数据的错误。DEBUG 命令的用法参见 LINGO|Debug 菜单命令。

### 6.2.5.2 GO

GO 命令用于编译并求解当前模型。当 LINGO 编译模型时,它将产生一个内部的可执行的模型版本,运行后可得到求解报告。

当 LINGO 完成模型的求解后,它将全部的结果报告显示到屏幕上。如果不想使用全部结果报告,在 GO 命令之前选用 TERSE 命令。

如果要把由 GO 命令产生的结果报告送到一个文件里,运行 GO 命令之前使用 DIVERT 命令。

参阅 SET 命令可了解 LINGO 求解器相关操作的各种参数设置信息。

### 6.2.5.3 NONZ

NONZ 或者 NONZEROS 命令用于显示当前模型的简短结果报告。NONZ 命令只显示非零变量和固定的约束条件(即松弛变量或者剩余变量为 0)。除了这点以外,NONZ 命令和 SOLUTION 命令是一样的。NONZ 命令的语法如下:

```
NONZ ['header_text'] [var_or_row_name]
```

如果忽略两个可选的参数,直接输入 NONZ 命令,就可以得到一个标准的 NONZ 结果报告,即 LINGO 将给出所有的非零变量和固定的约束条件的原始值和对偶值。LINGO 将对结果中所有的列给出标志。

第一个可选域 header_text 用于为结果显示一个标题。如果给出 header_text 参数,LINGO 只显示变量值,而忽略报告中的所有标记。

第二个可选域 var_or_row_name 是一个变量或者行的名称,如果给出该参数,报告中只显示所给的变量或者行。下面以 Chess Snackfoods 模型为例,说明用 NONZ 命令产生的几个结

**果报告：**

```
:TAKE CHESS.LNG
:TERSE
:GO
Global optimal solution found at step:        0
Objective value:                          2692.308
:! Generate a standard NONZ report
:NONZ
                     Variable      Value        Reduced Cost
                SUPPLY(PEANUTS)   750.0000      0.000000
                SUPPLY(CASHEWS)   250.0000      0.000000
                    PRICE(PAWN)   2.000000      0.000000
                  PRICE(KNIGHT)   3.000000      0.000000
                  PRICE(BISHOP)   4.000000      0.000000
                    PRICE(KING)   5.000000      0.000000
                 PRODUCE(PAWN)    769.2308      0.000000
                 PRODUCE(KING)    230.7692      0.000000
      FORMULA(PEANUTS,PAWN)       15.00000      0.000000
    FORMULA(PEANUTS,KNIGHT)       10.00000      0.000000
    FORMULA(PEANUTS,BISHOP)       6.000000      0.000000
      FORMULA(PEANUTS,KING)       2.000000      0.000000
      FORMULA(CASHEWS,PAWN)       1.000000      0.000000
    FORMULA(CASHEWS,KNIGHT)       6.000000      0.000000
    FORMULA(CASHEWS,BISHOP)       10.00000      0.000000
      FORMULA(CASHEWS,KING)       14.00000      0.000000
          Row    Slack or Surplus    Dual Price
            1    2692.308             1.000000
            2    0.000000             1.769231
            3    0.000000             5.461538
:! Generate a NONZ report for PRODUCE
:NONZ PRODUCE
              Variable      Value       Reduced Cost
         PRODUCE(PAWN)     769.2308      0.000000
         PRODUCE(KING)     230.7692      0.000000
:! Now add a header
:NONZ 'NONZERO PRODUCTION VALUES:' PRODUCE
NONZERO PRODUCTION VALUES:
     769.2308
     230.7692
```

### 6.2.5.4 RANGE

RANGE **命令用于在当前活动窗口为模型产生一个灵敏度分析报告。在默认状态下** Range

计算功能是不可用的,所以需要使用如下语句:

```
SET DUALCO 2;
```

因为 Range 计算需要花费一些时间,所以如果希望求解速度快,就不要使用 Range 计算功能。RANGE 命令的用法参见 LINGO|Range 菜单命令。

### 6.2.5.5 SOLU

SOLU 或者 SOLUTION 命令用于显示当前模型的求解结果报告。SOLU 命令的语法如下:

```
SOLU ['header_text'] [var_or_row_name]
```

一个标准的结果报告可以忽略两个可选参数,直接输入 SOLU 即可。这时,LINGO 将显示所有变量和行的原始值和对偶值,并且结果中所有的列都会给出标志。

第一个可选域 header_text 用于为结果报告显示一个标题。如果给出了 header_text 参数,LINGO 只显示结果值,忽略结果中所有的标志。

第二个可选域 var_or_row_name 表示一个变量或者约束条件的名称。如果给出了该参数,结果报告中只显示所给变量或者约束条件的信息。

仍然以 Chess Snackfoods 模型为例,运行 SOLU 命令后产生了以下几个结果报告:

```
:TAKE CHESS.LNG
:TERSE
:GO
Global optimal solution found at step:        0
Objective value:                          2692.308
:! Generate a standard SOLU report
:SOLU
                     Variable           Value        Reduced Cost
           SUPPLY(PEANUTS)         750.0000            0.0000000
           SUPPLY(CASHEWS)         250.0000            0.0000000
              PRICE(PAWN)          2.000000            0.0000000
            PRICE(KNIGHT)          3.000000            0.0000000
            PRICE(BISHOP)          4.000000            0.0000000
              PRICE(KING)          5.000000            0.0000000
            PRODUCE(PAWN)          769.2308            0.0000000
          PRODUCE(KNIGHT)          0.000000            0.1538461
          PRODUCE(BISHOP)          0.000000            0.7692297E-01
            PRODUCE(KING)          230.7692            0.0000000
    FORMULA(PEANUTS,PAWN)          15.00000            0.0000000
  FORMULA(PEANUTS,KNIGHT)          10.00000            0.0000000
  FORMULA(PEANUTS,BISHOP)          6.000000            0.0000000
    FORMULA(PEANUTS,KING)          2.000000            0.0000000
    FORMULA(CASHEWS,PAWN)          1.000000            0.0000000
```

```
FORMULA(CASHEWS,KNIGHT)     6.000000      0.0000000
FORMULA(CASHEWS,BISHOP)     10.00000      0.0000000
  FORMULA(CASHEWS,KING)     14.00000      0.0000000
          Row    Slack or Surplus      Dual Price
            1           2692.308        1.000000
            2           0.000000        1.769231
            3           0.000000        5.461538
 :! Generate a SOLU report for PRODUCE
 :SOLU PRODUCE
             Variable       Value     Reduced Cost
       PRODUCE(PAWN)       769.2308     0.0000000
     PRODUCE(KNIGHT)       0.000000     0.1538461
     PRODUCE(BISHOP)       0.000000     0.7692297E-01
       PRODUCE(KING)       230.7692     0.0000000

 :! Now add a header
 :SOLU 'PRODUCTION QUANTITIES' PRODUCE
PRODUCTION QUANTITIES
     769.2308
     0.000000
     0.000000
     230.7692
```

## 6.2.6 编辑命令

### 6.2.6.1 ALTER

ALTER 命令用于修改当前的模型。ALTER 命令的语法如下：

```
ALTER [line_number | line_range | ALL] 'old_string' new_string'
```

其中，line_number 是要修改的行的索引号；line_range 是要修改的行的范围；ALL 表示要修改模型中的所有行；old_string 代表被替换的字符串；new_string 表示要替换的新字符串。

以下用两个 ALTER 命令修改 knapsack 模型：

```
:TAKE ALTER.LNG
:LOOK ALL
 1] SETS:
 2]    THINGS/1..4/:VALUE, WEIGHT, X;
 3] ENDSETS
 4] DATA:
 5]    VALUE  =  8  6  4  3;
 6]    WEIGHT = 66 44 35 24;
```

```
  7] ENDDATA
  8]    MAX = @SUM(THINGS:VALUE * X);
  9]    @SUM(THINGS:WEIGHT * X) >= 100;
 10]    @FOR(THINGS:@BIN(X));
:! Change the direction of the constraint
:ALTER 9 '>=' <='
  9]    @SUM(THINGS:WEIGHT * X) <= 100;
:! Change 'THINGS' to 'ITEMS' in ALL rows
:ALTER ALL 'THINGS' 'ITEMS'
  2] ITEMS/1..4/:VALUE, WEIGHT, X;
  8]    MAX = @SUM(ITEMS:VALUE * X);
  9]    @SUM(ITEMS:WEIGHT * X) <= 100;
 10]    @FOR(ITEMS:@BIN(X));
:LOOK ALL
  1] SETS:
  2] ITEMS/1..4/:VALUE, WEIGHT, X;
  3] ENDSETS
  4] DATA:
  5]    VALUE  =  8  6  4  3;
  6]    WEIGHT = 66 44 35 24;
  7] ENDDATA
  8]    MAX = @SUM(ITEMS:VALUE * X);
  9]    @SUM(ITEMS:WEIGHT * X) <= 100;
 10]    @FOR(ITEMS:@BIN(X));
:
```

**注意：除了单引号以外，ALTER 命令也可以用双引号来划定替换文本的界限。**

#### 6.2.6.2 DELETE

DELETE 命令用于删除当前模型中的一行或者多行。DELETE 命令的语法如下：

```
DELETE [line_number | line_range|ALL]
```

其中，line_number 为要删除行的索引；line_range 为要删除行的范围；ALL 表示删除整个模型。

以下是 DELETE 命令的一些例子：

```
!删除模型中的第三行；
DELETE 3
!删除模型中的 2 到 10 行；
DEL 2 10
!删除整个模型；
DEL ALL
```

#### 6.2.6.3 EXTEND

EXTEND 命令用于在当前模型的末尾添加行。当输入 EXTEND 命令后,会出现问号提示,这时可以开始往模型中添加新的行。当输入完毕后,用 END 结束。

在下面的例子中,用 EXTEND 命令向一个小模型中添加新的约束条件:

```
:LOOK ALL
  1]MAX 20 * X + 30 * Y;
  2]X < = 50;
  3]Y < = 60;
  4]X + 2 * Y < = 120;
:!Use EXTEND to add another line
:EXTEND
?X > = 30;
?END
:LOOK ALL
  1]MAX 20 * X + 30 * Y;
  2]X < = 50;
  3]Y < = 60;
  4]X + 2 * Y < = 120;
  5]X > = 30;
:
```

### 6.2.7 交互参数命令

#### 6.2.7.1 PAGE

PAGE 命令用于设置屏幕显示的行数。PAGE 命令的语法如下:

```
PAGE n;
```

其中,n 是每页希望输出的行数。例如,PAGE 25 将使结果报告在显示 25 行时暂停,直到键入回车键后才接着显示下面的 25 行。PAGE 命令对于翻阅较长的报告很方便,以免结果超出屏幕顶部,影响阅读。

PAGE 的参数如果设为 0,则分页功能失去作用。一般情况下,如果不想间断命令脚本,可以在所有命令脚本的上部输入 PAGE 0 命令。

#### 6.2.7.2 PAUSE

PAUSE 命令用于暂停屏幕显示,直到键入回车键为止。如果在输入 PAUSE 命令的同一行中输入文本,这些文本会显示出来。这点对于在命令脚本中将提示信息传输给用户非常有用。

#### 6.2.7.3 TERSE

TERSE 命令用于阻止 LINGO 使用 GO 命令后产生的结果报告以自动显示的内容出现,而

是以概要模式输出结果报告。使用 TERSE 命令,需要使用 NONZ 或者 SOLU 命令查看结果报告。

一般情况下,当输出结果到电子表格或数据库时,都会输出总结报告,而当 LINGO 处于概要输出模式时,总结报告将不再输出。

执行 TERSE 命令后,LINGO 将一直处于概要输出模式,直到键入 VERBOSE 命令。

#### 6.2.7.4 VERBOSE

VERBOSE 命令用于终止 TERSE 命令的作用,使 LINGO 处于详细输出模式。详细输出模式是 LINGO 的默认模式,即模型的结果报告以自动显示的内容输出。在详细输出模式下,无论何时向电子表格和数据库进行输出操作,都会自动显示输出总结报告。

#### 6.2.7.5 WIDTH

WIDTH 命令用于限制终端输入和输出的宽度。WIDTH 命令的语法如下:

```
WIDTH n;
```

其中,n 是期望的最大宽度。n 值应在 64 ~ 200 之间,系统默认值是 76。

当 LINGO 生成结果报告时,输出行的宽度被限制在最大宽度之内,超出的部分将会被截去。

### 6.2.8 设置命令

#### 6.2.8.1 DBUID 和 DBPWD

DBUID 和 DBPWD 命令用于通过 @ODBC() 函数访问数据库时输入用户名和密码。用这两个命令输入的信息不会被永久保存。因此,在每一次访问数据库时都需要重新输入。这两个命令的语法如下:

```
DBUID my_user_id;
DBPWD my_password;
```

#### 6.2.8.2 FREEZE

FREEZE 命令用于将当前模型的配置保存到 LINGO 的配置文件中。当下一次运行时,LINGO 自动应用该配置。所有的配置信息都被保存到 LINGO 主目录下的 LINGO.CNF 文件中。该配置文件是一个文本文件,用户可以用文本编辑器打开它。通过 SET 命令设置的所有参数都可以通过 FREEZE 命令被保存。

当保存一个非默认配置时要注意,LINGO 下一次启动时,被保存的配置将生效。一些特定的参数配置会影响模型求解的方式,这种配置用于其他模型还可能导致错误的结果。为了恢复默认的配置,可用如下命令:

```
:SET DEFAULT
:FREEZE
```

6.2.8.3 SET

SET 命令用于修改 LINGO 的默认参数和设置。在 LINGO 中,所有用户的参数都可以通过 SET 命令来设置。SET 命令的语法如下:

```
SET parameter_name | parameter_index [parameter_value]
```

其中,parameter_name 是要设置的参数名;parameter_index 是要设置的参数索引;parameter_value 是新的参数值,如果它被忽略,LINGO 将显示指定参数的当前值。

## 6.2.9 其他命令

6.2.9.1 ! (注释)

在一个命令中插入一个惊叹号,其后面的内容将被求解器忽略。该命令常用于为模型添加注释语句。

6.2.9.2 NEWPW

NEWPW 命令用于设置访问 LINGO 版本的一个新的密码。这个命令可用来更新该版本 LINGO 的性能。要了解更多的 LINGO 更新和增强功能,可访问 LINGO Systems 网站,地址为 www.lindo.com。

要了解目前使用版本的功能和特征的信息可使用 HELP 命令。

6.2.9.3 QUIT

QUIT 命令用于关闭 LINGO。在关闭 LINGO 之前,一定要确保已经保存了修改后的模型。

6.2.9.4 TIME

TIME 命令用于显示从 LINGO 开始求解到现在为止所用的时间。下面的例子中说明了使用 TIME 命令的语法:

```
:TIME
Cumulative HR:MIN:SEC = 2:22:39.54;
```

# 第 7 章　LINGO 的运算符和函数

作为非常方便的建模工具，LINGO 提供了许多函数和运算符。这些运算符和函数可以分为以下几类：

1）标准运算符——算术、逻辑和关系运算符，例如 + 、- 、= 、< = 等；

2）数学函数——三角函数和一般数学函数；

3）金融函数——常用金融函数，常常用于确定资金的现值；

4）概率函数——用于确定概率的范围和统计结果，例如 Poisson 和 Erlang 队列函数等；

5）变量限定函数——用于确定变量的取值范围，其用法参见第 3 章；

6）集合处理函数——操作集合的函数；

7）集合循环函数——对集合中所有元素完成某种运算的函数，例如计算集合元素的和、最大值或最小值等，其用法参见 2.5 节中的内容；

8）输入和输出函数——用于创建与外界数据源连接的函数。

## 7.1　标准运算符

LINGO 提供了算术运算符、逻辑运算符、关系运算符三种标准运算符。

### 7.1.1　算术运算符

算术运算符较为简单，用于数字间的操作。LINGO 有五个二元的算术运算符，如表 7-1 所示。

LINGO 中唯一一个一元算术运算符是取负符（-）。表 7-1 中的算术运算符的优先顺序如表 7-2 所示。

表 7-1　算术运算符及功能

| 算术运算符 | 功能 |
|---|---|
| ^ | 求幂 |
| * | 乘 |
| / | 除 |
| + | 加 |
| - | 减 |

表 7-2　算术运算符优先级

| 优先级 | 算术运算符 |
|---|---|
| 最高 | -（取负） |
| ↓ | ^ |
| ↓ | * / |
| 最低 | + - |

具有最高优先权的运算符要先计算，并且计算顺序是从左到右。但是，运算符的计算顺序能够被括号控制，即 LINGO 首先计算最内部括号里的表达式的值，然后计算其外的。

## 7.1.2 逻辑运算符

在 LINGO 中,逻辑运算符主要用在集合循环函数的条件表达式中,用于控制函数中应该包含或不包含的集合元素。此外,在建立集合元素条件时也起一定作用。

逻辑运算的结果是 TRUE (真) 或者 FALSE (假)。LINGO 中,一般用数值 1 代表 TRUE,0 代表 FALSE。对于变量,当且仅当变量的值等于 0 时,它才被认为是 FALSE。因此,例如 1、7、- 1 和 0.123 4 等都代表 TRUE。

LINGO 有 9 个逻辑运算符,除了 # NOT # 运算符是一元的以外,其他都是二元的。表 7-3 列举了 LINGO 逻辑运算符和各自的返回值。

表 7-3 逻辑运算符及其返回值

| 逻辑运算符 | 返回值 |
|---|---|
| # NOT # | 如果右边的操作数是 FALSE,则返回值为 TRUE,否则为 FALSE |
| # EQ # | 如果两个操作数相等,返回值为 TRUE,否则为 FALSE |
| # NE # | 如果两个操作数不相等,返回值为 TRUE,否则为 FALSE |
| # GT # | 如果左边的操作数大于右边的操作数,返回值为 TRUE,否则为 FALSE |
| # GE # | 如果左边的操作数大于或者等于右边的操作数,返回值为 TRUE,否则为 FALSE |
| # LT # | 如果左边的操作数小于右边的操作数,返回值为 TRUE,否则为 FALSE |
| # LE # | 如果左边的操作数小于或者等于右边的操作数,返回值为 TRUE,否则为 FALSE |
| # AND # | 只有两个事件都为 TRUE 时,返回值才为 TRUE,否则为 FALSE |
| # OR # | 只有两个事件都为 FALSE 时,返回值才为 FALSE,否则为 TRUE |

逻辑运算符的优先级如表 7-4 所示。

表 7-4 逻辑运算符优先级

| 优先级 | 逻辑运算符 |
|---|---|
| 最高 | # NOT # |
| ↓ | # EQ #、# NE #、# GT #、# GE #、# LT #、# LE # |
| 最低 | # AND #、# OR # |

## 7.1.3 关系运算符

在 LINGO 里,模型中的关系运算符用于指定表达式的左边是否等于、小于等于或大于等于表达式的右边,所以关系运算符主要用来建立模型的约束条件。与逻辑运算符不同,关系运算符要求模型的结果必须符合关系运算符的要求,而逻辑运算符的结果表示约束条件是否被满足。关系运算符在所有的运算符中优先权最低。表 7-5 列出了 LINGO 提供的三个关系运算符。

在 LINGO 中也可以用“ < ”代替“ < = ”,用“ > ”代替“ > = ”。

注意：LINGO 不直接支持严格小于或大于的关系运算符。一般情况下，找不到具有这样一个特征的表达式。但是，如果想要 A 严格小于 B，即 A < B，可以将这个数学式转换成以下等价的数学式：

$$A + \varepsilon \leqslant B$$

式中含有一个很小的常量 ε，其值取决于 A 必须比 B 小多少，这样才能够被认为是不相等的。

表 7-5 关系运算符及义

| 关系运算符 | 解释 |
|---|---|
| = | 左右两边必须相等 |
| < = | 左边必须小于或等于右边 |
| > = | 左边必须大于或等于右边 |

### 7.1.4 运算符优先级

算术运算符、逻辑运算符和关系运算符的优先级顺序列于表 7-6 中。

表 7-6 运算符优先级

| 优先级 | 运算符 |
|---|---|
| 最高级 | # NOT # - (否定) |
| | |
| | * / |
| | + - |
| | # EQ # # NE # # GT # # GE # # LT # # LE # |
| | # AND # # OR # |
| 最低级 | < = = > = |

## 7.2 数学函数

LINGO 提供了一些标准的数学函数，这些函数针对标量自变量返回一个结果。这些函数的形式和意义如表 7-7 所示。

表 7-7 数学函数及意义

| 函数 | 意义 |
|---|---|
| @ABS (X) | 返回 X 的绝对值 |
| @COS (X) | 返回 X 的余弦函数值，X 是角的弧度值 |
| @EXP (X) | 返回 e (2.718 281...) 的 X 次方的值 |
| @FLOOR (X) | 返回 X 的整数部分。如果 X ⩾ 0，@FLOOR 返回小于等于 X 的最大整数；如果 X < 0，@FLOOR 返回大于等于 X 的最小整数 |
| @LGM (X) | 返回 X 的 gamma 函数值的自然对数，即 log [(X - 1) !]，通过线性插值可以获得 X 为非整数值的情况 |

续表

| 函数 | 意义 |
| --- | --- |
| @LOG(X) | 返回 X 的自然对数 |
| @SIGN(X) | 如果 X < 0,返回 - 1 否则,返回 + 1 |
| @SIN(X) | 返回 X 的正弦函数值,X 是角的弧度值 |
| @SMAX(X1,X2,…,XN) | 返回数列 X1、X2、…、XN 中的最大值 |
| @SMIN(X1,X2,…,XN) | 返回数列 X1、X2、…、XN 中的最小值 |
| @TAN(X) | 返回 X 的正切函数值,X 是角的弧度值 |

## 7.3 金融函数

LINGO 目前提供两个金融函数,一个用于计算年金的现值,另一个用于计算一次性付清总额的现值。

1. @FPA(I,N)

@FPA(I,N)函数用于计算年金为 1 元的现值,即从现在开始的 N 个时期内每个时期支付 1 元钱。利率 I 不是百分数,而是描述利率的一个小数(例如,用 0.1 代表 10%)。结果乘以 X 就可以得到年金为 X 元的现值。

2. @FPL(I,N)

@FPL(I,N)函数用于计算从现在开始的第 N 个时期一次付清 1 元钱的现值,利率 I 同样是一个小数。结果乘以 X 就可以得到一次付清 X 元的现值。

## 7.4 概率函数

1. @PBN(P,N,X)

@PBN(P,N,X)函数用于计算累积二项概率,该函数返回包含 N 个个体的样本中有 X 个或者更少个体不合格的概率,样本取自个体不合格率是 P 的总体。该函数通过线性插值可以扩展到 X 是非整数值的情况。

2. @PCX(N,X)

@PCX(N,X)是自由度为 N 的 $\chi^2$ 分布的累积分布函数。该函数返回符合 $\chi^2$ 分布的观察值小于等于 X 的概率。

3. @PEB(A,X)

@PEB(A,X)函数用于计算服务系统的 Erlang 忙碌概率。服务系统有 X 个服务台,到达负荷为 A(A 是单位时间到达的顾客期望数量乘以每个顾客接受服务的期望时间),允许无限排队等待。函数结果可以解释为所有服务台都是忙碌状态的时间比例,或者解释为必须排队等待的顾客比例。该函数通过线性插值可以扩展到 X 是非整数值的情况。

4. @PEL(A,X)

@PEL(A,X)函数用于计算服务系统的 Erlang 损失率。服务系统有 X 个服务台,到达负荷

为 A,不允许排队等待。@PEL 函数的结果可以解释为所有服务台都是忙碌状态的时间比例,也可以解释为顾客到达时因为所有服务台忙碌而损失的顾客比例。该函数通过线性插值可以扩展到 X 是非整数值的情况。

5. @PFD(N, D, X)

@PFD(N,D,X)是分子自由度为 N、分母自由度为 D 的 F 分布的累积分布函数。该函数返回符合 F 分布的观察值小于等于 X 的概率。

6. @PFS(A, X, C)

@PFS(A,X,C)是系统中等待或正在接受服务的顾客人数的期望值。该服务系统是资源有限的泊松服务系统,系统中有平行的 X 个服务台、C 个顾客和有限的工作负荷 A。该函数通过线性插值可以扩展到 X 是非整数值的情况。有限的工作负荷 A 是顾客数量乘以服务的平均时间再除以服务的平均时间。

7. @PHG(POP, G, N, X)

@PHG(POP,G,N,X)是累积超几何分布概率函数。该函数返回包含 N 个个体的样本中有 X 个或者更少个体合格的概率,并且假设该样本不再重新放回总体。其中,样本来自的总体大小为 POP,合格率是 G。该函数通过线性插值可以扩展到 POP、G、N 和 X 是非整数值的情况。

8. @PPL(A, X)

@PPL(A,X)是泊松分布的线性损失函数。该函数返回 MAX(0, Z - X)的期望值,其中 Z 是服从均值为 A 的泊松分布的随机变量。

9. @PPS(A, X)

@PPS(A,X)是累积泊松概率函数。该函数返回均值为 A 的泊松随机变量小于等于 X 的概率。该函数通过线性插值可以扩展到 X 是非整数值的情况。

10. @PSL(X)

@PSL(X)是单位正态分布的线性损失函数。该函数返回 MAX(0,Z - X)的期望值,其中 Z 是服从标准正态分布的随机变量。

11. @PSN(X)

@PSN(X)是累积标准正态概率分布函数。标准正态随机变量的均值为 0.0,标准偏差为 1.0(钟形曲线,以原点为中心)。该函数返回的横坐标在 X 左边的曲线下方的面积值。

12. @PTD(N, X)

@PTD(N,X)是自由度为 N 的 t 分布的累积分布函数。该函数返回符合 t 分布的观察值小于等于 X 的概率。

13. @QRAND(SEED)

该函数用于产生(0,1)区间的拟随机标准化数的序列。@QRAND 函数只能应用在模型的数据域中。用其产生的拟随机数可以完全填充一个集合的属性。一般情况下,属性值是一个二维表格,有 m 行、n 个变量。m 代表情景或者试验的次数,n 代表每个情景或者试验中随机变量的个数。每行中的数字独立分布,所有行中的数字分布完全一致,这些数据由分层抽样产生。

例如,设 m = 4,n = 2。虽然数据是随机的,但总会有一行数据值全部在(0,0.5)之间,一行数据值全部在(0.5,1)之间,另外两行中一个数小于 0.5,一个数大于 0.5。如果希望生成 8 个

普通的随机数据，用函数@QRAND(1,8)比用函数@QRAND(4,2)好。在下面的例子中说明了@QRAND函数的用法：

```
MODEL:
  DATA:
    M = 4;
    N = 2;
    SEED = 1234567;
  ENDDATA
  SETS:
    ROWS/1..M/;
    COLS/1..N/;
    TABLE(ROWS, COLS):X;
  ENDSETS
  DATA:
    X = @QRAND(SEED);
  ENDDATA
END
```

如果没有给@QRAND指定一个SEED值，LINGO会根据系统时钟建立一个SEED值。

14. @RAND(SEED)

该函数返回(0,1)之间的一个伪随机数，其值由SEED决定。

## 7.5 集合处理函数

LINGO提供了一些处理集合的函数。其中，@IN函数用于确定一个元素是否被包含在集合中；@INDEX函数返回一个基本集合的元素在集合里的索引值；@SIZE函数返回集合元素的总个数；@WRAP函数用于限制变量的索引在一定范围内循环。

1. @IN(set_name, primitive_index_1 [, primitive_index_2 ...])

如果根据基本集合索引值(primitive_index_1, primitive_index_2, ...)所表示的集合元素是在集合set_name里，则@IN函数返回TRUE。正如下面的例子所示，@IN函数对生成子集合很有用。

为了根据已关闭工厂的子集合获得开工工厂的集合，可以建立下面的集合域：

```
SETS:
  PLANTS/SEATTLE, DENVER,
  CHICAGO, ATLANTA/:;
  CLOSED(PLANTS)/DENVER/:;
  OPEN(PLANTS)|
   #NOT# @IN(CLOSED, &1):;
ENDSETS
```

集合 OPEN 衍生于集合 PLANTS,方法是采用含有 @IN 函数的元素判别条件,只允许那些不属于集合 CLOSED 的元素进入集合 OPEN。

下面的例子说明如何确定集合元素 (B,Y) 是否属于衍生集合 S3。因为 (B,Y) 的确是 S3 的一个元素,所以 X 将等于 1。注意:为了能够得到基本集合中的元素 B 和 Y 的索引值,使用了 @INDEX 函数。例子如下:

```
SETS:
  S1 / A B C/:;
  S2 / X Y Z/:;
  S3 (S1, S2) / A,X A,Z B,Y C,X/:;
ENDSETS
  X = @IN(S3, @INDEX(S1, B), @INDEX(S2, Y));
```

2. @INDEX ( [set _ name,] primitive _ set _ element)

@INDEX 函数用于返回基本集合 set _ name 中的元素 primitive _ set _ element 的索引值。如果集合名 set _ name 被忽略,则 LINGO 返回能找到的第一个名为 primitive _ set _ element 的基本集合元素的索引值。如果 LINGO 没有找到 primitive _ set _ element,则会出现错误信息。

如果一个模型的集合元素来自外部的数据源,建模者可能很难控制这个数据源。当使用 @INDEX 函数时,如果不指定集合名常常会引起混乱。考察下面的集合域:

```
SETS:
  GIRLS /DEBBIE, SUE, ALICE/;
  BOYS /BOB, JOE, SUE, FRED/;
ENDSETS
```

现在,假使希望在集合 BOYS 中找到男孩 SUE 的索引。这个索引值应该是 3。如果简单地用 @INDEX (SUE) 函数,得到的值是 2 而不是 3。这是由于 LINGO 首先在集合 GIRLS 里发现了 SUE。为了获得正确的结果,必须指定 BOYS 集合作为函数中的参数,即 @INDEX ( BOYS, SUE)。所以应该在 @INDEX 函数中指定集合名。

3. @WRAP ( INDEX, LIMIT)

@WRAP 函数可以转换集合两端的索引,在集合的另一端继续索引。也就是说,在集合循环函数里,当达到集合的最后一个 (或者第一个) 元素后,可以用 @WRAP 函数把索引转到集合的第一个 (或者最后一个) 元素。在循环、多期计划模型中,这是一个很有实用价值的函数。

以数学形式表达,@WRAP 函数返回 J,J = INDEX - K × LIMIT,其中 K 是使 J 落在 1 到 LIMIT 之间的整数。即 @WRAP 函数可以减少或者增加整数倍的 LIMIT 给 INDEX,直到其落在 1 到 LIMIT 范围之间。

4. @SIZE ( set _ name)

@SIZE 函数返回集合 set _ name 所含元素的个数。使用 @SIZE 函数可以明确知道模型中一个集合的大小。这对于数据的独立和维护很有帮助。

## 7.6 输入/输出函数

输入和输出函数用于模型与外界数据源的连接,例如文本文件、数据库和电子数据表等。

目前，LINGO 提供下列几种输入和输出函数。

1. @DUAL ( variable _ or _ row _ name)

在一个输出语句里，@DUAL 函数用于输出变量或行的对偶值。例如下面的数据域：

```
DATA:
  @TEXT('C:\RESULTS\OUTPUT.TXT') = X, @DUAL(X);
ENDDATA
```

当求解模型时，变量 X 的值和它的降低成本值将输出到 C：h RESULTS h OUTPUT. TXT 文件中。如果函数的参数是一个行名，则所有生成行的对偶价格同行名一起被输出。借助输出语句左边的函数，可以将结果输出到一个文件、电子表格、数据库或指定的内存区域。

2. @FILE ( ' filename')

@FILE 函数用于读取外部文本文件中的数据，它可以放在模型中的任何位置。其中，filename 为文本文件的名称。在将集合域和数据域里的数据合并到文本文件中时，这个函数非常有用。

在模型中执行这个函数时，LINGO 将连续从这个文件里读取数据直到文件尾或一个 LINGO 记录结束符号 (~)。在同一个模型里再次执行 @FILE 函数时，则 LINGO 从上次离开的地方开始读取数据。在 LINGO 中，不支持 @FILE 函数的嵌入调用 (在一个文件里嵌入一个 @FILE函数，而这个文件又被 @FILE 函数调用)。

3. @ODBC ( [data _ source' [, ' table _ name' [, ' col _ 1' [, ' col _ 2' ...]]]])

@ODBC 函数用于打开 LINGO 和数据库之间的 ODBC 连接。该函数可以放在集合域中，用于从数据库中导入集合元素，或放在数据域里，用于导入数据或导出求解结果。data _ source 是在 ODBC 管理器中注册的 ODBC 数据源的名称。table _ name 是希望连接的数据源中数据表的名称。col _ I 是在 table _ name 表中希望连接的列名。

4. @OLE ( ' spreadsheet _ file' [, range _ name _ list])

@OLE 函数是一个接口函数，用于通过 OLE 技术从 EXCEL 数据表中读入数据并将结果输出到 EXCEL 中。OLE 传输直接利用内存，不使用任何中介文件。当使用 @OLE 函数进行输出时，LINGO 首先加载 EXCEL 程序，并命令 EXCEL 加载需要的数据表，最后发送包含结果的数据区域到工作表中。使用 @OLE 函数，必须有 EXCEL 5.0 或者更新的版本，并且 @OLE 函数只能应用在数据域或者集合域中。@OLE 函数只能输出二维的数据区域，不能输出三维的数据区域或者间断的区域。

spreadsheet _ file 参数是要连接的电子表格的文件名称。range _ name _ list 是要连接的数据区域的名称列表。

5. @POINTER ( N)

@POINTER 函数直接用于 Windows 下的 LINGO 动态连接库 (DLL)，从共享的内存区域传输数据。

6. @RANGED ( variable _ or _ row _ name)

@RANGED 函数用于输出特定变量在目标函数中系数的允许减少量，或特定行右边量的允许减少量。例如下面的数据域：

```
DATA:
  @TEXT('C:\RESULTS\OUTPUT.TXT') = X, @RANGED(X);
ENDDATA
```

当求解模型时,变量 X 的值和它在目标函数中的系数的允许减少量将被写入文件 C:\RESULTS\OUTPUT.TXT 中。

7. @RANGEU(variable_or_row_name)

@RANGEU 函数用于输出特定变量在目标函数中系数的允许增加量或特定行右边量的允许增加量。该函数的用法参见 @RANGED 函数。

8. @STATUS()

@STATUS 函数通过使用表 7-8 中的代码输出求解的最终状态。

表 7-8 @STATUS()函数的代码及意义

| 代码 | 说明 |
|---|---|
| 0 | 找到全局最优解 |
| 1 | 无可行解 |
| 2 | 目标函数无界 |
| 3 | 求解过程失败 |
| 4 | 保留 |
| 5 | 不可行或者无界,前处理程序确定模型是不可行的或无界时,关闭 presolving 功能并且重新求解以确定到底是哪种情况 |
| 6 | 局部最优解,尽管一个更好的解可能存在,但是已找到一个局部最优解 |
| 7 | 局部不可行,尽管可行解可能存在,但是 LINGO 找不到 |
| 8 | 求解中断,达到了目标函数的中断水平 |
| 9 | 数值错误,某个约束上的未定义的数学操作导致求解停止 |

一般情况下,如果 @STATUS 函数没有返回代码 0,则结果没有价值而且不可信。@STATUS 函数只能用于数据域里的输出。

9. @TEXT([filename'])

@TEXT 函数用于将模型数据域中的结果输出到文本文件。filename 是输出文件的名称。如果'filename'被忽略,数据将被输出到标准的输出装置(大多数情况下指屏幕)。

## 7.7 其他函数

1. @IF(logical_condition, true_result, false_result)

@IF 函数首先判断 logical_condition 是否为真,如果是,则返回 true_result;否则返回 false_result。下面的例子中,用 @IF 函数计算固定生产成本:

```
MIN = COST;
COST = XCOST + YCOST;
```

```
XCOST = @IF(X #GT# 0,100,0) + 2 * X;
YCOST = @IF(Y #GT# 0,60,0) + 3 * Y;
X + Y >= 30;
```

在模型中,生产两种产品 X 和 Y,在产品总量不小于 30 的条件下,计算使总成本最小的产品产量。生产 X 时,固定成本是 100,可变成本是 2;生产 Y 时,则分别是 60 和 3。本模型应用 @IF 函数来判断产品是否被生产,以便计算相关的固定成本,即判断产品的产量是否大于 0,如果是,则函数返回固定成本,否则返回 0。

如果没有 @IF 函数的帮助,固定成本模型还可以调用一些使用二进制整数变量的窍门,但没有应用 @IF 函数构建的模型那样直观。

需要注意的是,@IF 函数不是一个线性的函数,最多只是一个分段线性函数。在上面的例子中,@IF 函数就是分段线性的,在原点处有一个不连续的断点。而 LINGO 模型最好是线性的,非线性的模型最好是连续的。显然,@IF 函数违反了这两个条件。因此,包含 @IF 函数的模型在解决全局最优化问题时会很困难。幸运的是,LINGO 有两个选项——线性化和全局最优化——可以克服这个问题。在不使用线性化和全局最优化方法的情况下,得到如下结果:

```
Local optimal solution found at iteration:  42
Objective value:  160.0000
     Variable           Value
      COST           160.0000
      XCOST          160.0000
      YCOST          0.000000
          X          30.00000
          Y          0.000000
```

结果显示只生产 X 的总成本为 160。显然,这仅是一个局部最优解,因为如果仅生产 Y 而不生产 X,总成本仅为 150。为了找到全局最优解,必须在 LINGO 中采用线性化和全局最优化的方法。

简要地说,线性化就是将非线性的模型转化成在数学上等价的线性模型。这样做有两个原因:其一,也是最重要的,线性模型总能找到全局最优解;其二,线性模型的求解速度比等价的非线性模型更快。但是,不是所有的非线性模型都能转化成等价的线性模型,在这种情况下,线性化将没有作用。上例中的模型能完全被线性化。为了激活线性化的选项,运行 LINGO|Options 命令,设置 General Solver 表中的 Linearization Degree 为 High。

全局最优化(Global Optimization)把模型分解成一系列较小的局部模型。这一系列的局部模型求解完毕后,全局最优解也就能确定了。为了激活全局最优化,运行 LINGO|Options 命令,选择 Global Solver 页之后点击 Global Solver 复选框。注意:全局求解方法是一个附加的 LINGO 选项,所以全局求解的功能在一些版本中不能使用。

无论使用线性化选项还是全局最优化方法,LINGO 都可以得到真正的全局最优解:

```
Global optimal solution found at iteration:  6
Objective value:  150.0000
    Variable           Value
```

```
COST            150.0000
XCOST           0.000000
YCOST           150.0000
  X             0.000000
  Y             30.00000
```

2. @WARN('text', logical_condition)

@WARN 函数用于判断逻辑条件 logical_condition 是否得到满足,如果是,则显示'text'的警告内容。这对于检查模型数据的合理性很有用。在下面的例子中,如果用户输入一个负利率,则会显示"INVALID INTEREST RATE"。

```
!A model of a home mortgage;
DATA:
!Prompt the user for the interest
 rate, years, and value of mortgage.
 We will compute the monthly payment;
   YRATE  = ?;
   YEARS  = ?;
   LUMP   = ?;
ENDDATA
!Number of monthly payment;
 MONTHS = YEARS * 12;
!Monthly interest rate;
 (1 + MRATE) ^ 12  =  1 + YRATE;
!Solve next line for monthly payment;
 LUMP = PAYMENT * @FPA(MRATE, MONTHS);
!Warn them if interest rate is negative
 @WARN('INVALID INTEREST RATE',
   YRATE #LT# 0);
```

3. @USER(user_determined_arguments)

@USER 函数的用法相对比较复杂,读者可以参考 11.2 节中的内容。

# 第 8 章　与外部文件的接口

在多数情况下,模型中所需的原始数据都保存在文本文件、电子数据表或数据库中。这样,如果在 LINGO 模型文件中进行数据维护工作将会非常麻烦而且效率很低,因此需要把数据从外部导入到 LINGO 模型中。另外,如果不能把 LINGO 的求解结果导出到其他应用软件中,LINGO 的应用范围就会变得很狭窄。基于上述原因,LINGO 提供了很多方法帮助用户实现数据在 LINGO 中的导入与导出。本章首先阐述如何通过 ASCII 文本将数据从 LINGO 中导入和导出,然后介绍了如何进行 LINGO 与电子数据表的连接等操作,最后阐述了怎样使用数据库来维护模型中的数据。

## 8.1　利用复制和粘贴命令传输数据

在 Windows 操作系统下,将数据从应用软件中导入和导出的最简单直接的方法是利用复制和粘贴命令。支持复制命令的应用软件可以将信息保存到系统剪贴板中,支持粘贴命令的应用软件可以将信息从系统剪贴板中导出。因此,这两个命令为从一个应用软件到另一个应用软件移动数量不大的数据提供了简单有效的方法。

### 8.1.1　从 Excel 中导入数据

任何支持复制命令的应用软件中的数据都可以导入到 LINGO 中。为了阐明这个问题,下面举例说明如何将数据从 Excel 电子数据表导入到 LINGO 模型中。这里利用第 2 章中员工安排模型的例子,在 DATA 数据部分作了一些改动,即省略了 REQUIRED 属性的赋值。模型如下:

```
SETS:
  DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
DATA:
  REQUIRED = ;
ENDDATA
MIN = @SUM(DAYS(I):START(I));
@FOR(DAYS(J):
  @SUM(DAYS(I) | I #LE# 5:
    START(@WRAP(J - I + 1, 7)))
      >= REQUIRED(J)
);
```

假设员工需求数据在 Excel 中通过图 8-1 所示形式保存。

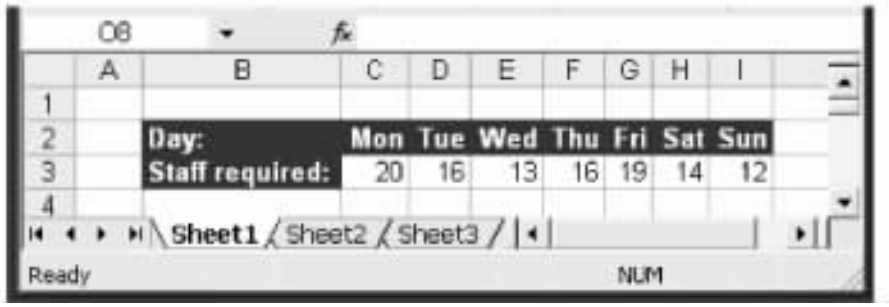



图 8-1　员工需求数据表

在将职员需求数据从 Excel 电子数据表粘贴到 LINGO 模型之前，应该完成以下步骤：

1) 将光标放于 C3 单元格，按下鼠标左键并拖曳到 I3 单元格，释放鼠标左键，这样就选中了从 C3 到 I3 范围的数据；

2) 选择 Excel 中“编辑”菜单的“复制”命令；

3) 单击 LINGO 模型窗口；

4) 将光标直接放于语句“REQUIRED = ”的右边；

5) 选择 LINGO 中“编辑”菜单的“粘贴”命令。

这样数据就会以如下的形式出现在 LINGO 模型中：

```
DATA:
  REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
```

还可以根据个人喜好利用 LINGO 中的“编辑|选择字体”命令调整数据的字体。通过上述操作，模型中就有了它所需要的数据并且能够运行。另外，LINGO 也可以将数据从 Excel 中直接导入，具体内容请参见以后的章节。

### 8.1.2　将数据导出到 Word 文件中

假设上面的员工安排模型已经运行并得出结果，LINGO 将显示一个包含有求解结果报告的新窗口。现在，若希望能够在 Word 中得到该模型求解结果的副本，可以通过以下步骤实现：

1) 单击 LINGO 中的 Solution Report Window，弹出结果报告窗口；

2) 选择 LINGO 中的 Edit|Select All (编辑|选中全部) 命令，即可选中结果报告窗口中的所有文本；

3) 选择 LINGO 中的 Edit| Copy (编辑|复制) 命令，将求解结果复制到剪贴板中；

4) 单击包含有报告内容的 Word 窗口；

5) 选中 Word 中的 Edit|Paste (编辑|粘贴) 命令，将求解结果从剪贴板导出到 Word 中。

## 8.2　文本文件接口函数

LINGO 有许多用于执行导入和导出数据功能的接口函数，负责处理与文本文件、电子数据

表软件和数据库之间的接口问题，LINGO 中还有能够实现在其他应用软件交换数据的接口。这部分将重点讨论 LINGO 怎样利用 @FILE 函数和 @TEXT 函数与文本文件进行连接。利用 @FILE函数可以将外部文本文件中的数据导入到 LINGO 模型中，而利用 @TEXT 函数可以将模型的求解结果导出到文本文件中。

## 8.2.1 利用 @FILE 函数导入数据

LINGO 中的 @FILE 函数可以放在模型的任何位置，用于将外部文本文件中的数据导入到 LINGO 模型中。@FILE 函数的语法是：

```
@FILE('filename')
```

其中，filename 是包含 LINGO 所需数据的文件名称。当模型中用到这个函数时，LINGO 将连续不断地从这个文件中读取文本直到遇到文件结束标志或 LINGO 的记录结束标志（~）。对于该模型中后来又出现的并且使用相同 filename 的 @FILE 函数，LINGO 将从它上次读数的停止点处继续从该文件中读取文本。@FILE 函数不允许嵌套使用。

## 8.2.2 在运输模型中使用 @FILE 函数

下面将通过改进的 Wireless Widgets 运输模型说明在 LINGO 中如何使用 @FILE 函数。1.4 节中该模型的形式如下：

```
! 一个包含 6 个 WAREHOUSES (仓库) 和 8 个 VENDORS (销售商) 的运输问题;
SETS:
  WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/ : CAPACITY;
  VENDORS / V1 V2 V3 V4 V5 V6 V7 V8/    : DEMAND;
  LINKS (WAREHOUSES, VENDORS) : COST, VOLUME;
ENDSETS
! 目标;
  MIN = @SUM(LINKS(I, J):
    COST(I, J) * VOLUME(I, J));
! 需求量约束;
  @FOR(VENDORS(J):
    @SUM(WAREHOUSES(I): VOLUME(I, J)) =
      DEMAND(J));
! 供应量约束;
  @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J): VOLUME(I, J)) < =
      CAPACITY(I));
! 数据部分;
DATA:
  CAPACITY = 60 55 51 43 41 52;
  DEMAND = 35 37 22 32 41 32 43 38;
  COST = 6 2 6 7 4 2 5 9
```

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

可见,模型中有两处出现了数据。首先,在集合域中有仓库 WAREHOUSES 和销售商 VENDORS 的列表 其次,在数据 DATA 部分有关于供应量 CAPACITY、需求量 DEMAND 和运输成本 COST 的数据。

为了将数据从模型中完全分离,可以将数据转移到一个外部文本文件中并修改这个模型。在本节内容中,LINGO 通过 @FILE 函数将数据从文本文件中提取出来。以下修改后的模型已经省略了所有的数据,有关改动都已用黑体字标出。程序如下:

```
SETS:
  WAREHOUSES / @FILE('WIDGETS2.LDT')/: CAPACITY;
  VENDORS / @FILE('WIDGETS2.LDT')/    : DEMAND;
  LINKS(WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
  MIN = @SUM(LINKS(I, J):
    COST(I, J) * VOLUME(I, J));
  @FOR(VENDORS(J):
    @SUM(WAREHOUSES(I): VOLUME(I, J)) =
    DEMAND(J));
  @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J): VOLUME(I, J)) < =
      CAPACITY(I));
DATA:
  CAPACITY = @FILE('WIDGETS2.LDT');
  DEMAND = @FILE('WIDGETS2.LDT');
  COST = @FILE('WIDGETS2.LDT');
ENDDATA
```

这时,该模型只从 WIDGETS2.LDT 文件中提取数据。该文本文件中的数据内容如下:

```
!仓库 WAREHOUSES 的成员列表;
WH1 WH2 WH3 WH4 WH5 WH6 ~
!销售商 VENDORS 的成员列表;
V1 V2 V3 V4 V5 V6 V7 V8 ~
!仓库 WAREHOUSES 的供应量数据;
60 55 51 43 41 52 ~
!销售商 VENDORS 的需求量数据;
35 37 22 32 41 32 43 38 ~
!单位运费;
6 2 6 7 4 2 5 9
```

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

按照习惯,用.LDT作为所有LINGO数据文件的扩展名。

在数据文件中记录结束标志(~)之间的部分称作记录。如果一个被LINGO读取的文件中没有记录结束标志(~),那么LINGO会以一条记录读取所有的文件内容。除了记录结束标志(~)之外,文本内容和它们在原来模型中的位置要一致。

在LINGO模型中应用@FILE函数调用文件数据时,记录结束标志(~)是按以下方式发挥作用的:当第一次调用@FILE函数时,打开WIDGETS2.LDT文件并且读取第一条记录,然后关闭文件,第二次调用时则读取第二条记录并关闭文件,依此类推。

文件中的最后一条记录后面不需要记录结束标志(~)。当LINGO遇到文件末尾时,它读取最后一条记录并关闭该文件。如果在文件的最后一条记录后有记录结束标志(~),那么在模型求解完成之前,LINGO都不能关闭该文件,这将导致一个模型中同时打开多个文件而超过允许同时打开文件数量的问题。

当使用@FILE函数时,应该一边考虑记录的内容,一边用@FILE函数代替模型中的相应内容。这样就能够把一个完整的陈述语句、部分的陈述语句或整个陈述语句序列放进一条记录中。例如,上例中WIDGETS2.LDT文件的头两条记录是:

```
!仓库 WAREHOUSES 的成员列表;
WH1 WH2 WH3 WH4 WH5 WH6 ~
!销售商 VENDORS 的成员列表;
V1 V2 V3 V4 V5 V6 V7 V8 ~
```

在LINGO模型的集合域部分,@FILE函数用如下的两条语句调用WIDGETS2.LDT文件里的头两条记录:

```
WAREHOUSES / @FILE('WIDGETS2.LDT')/:CAPACITY;
VENDORS / @FILE('WIDGETS2.LDT')/    :DEMAND;
```

@File函数调用的最终结果如下:

```
WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/:CAPACITY;
VENDORS / V1 V2 V3 V4 V5 V6 V7 V8/    :DEMAND;
```

注意 数据文本文件中的注释是被忽略的,而且在一个模型中最多能够同时包含16个数据文本文件。

### 8.2.3 利用@TEXT函数导出数据

@TEXT接口函数用于将模型的求解结果导出到文本文件中,@TEXT函数还可以将LINGO中集合域成员和属性值导出到文本文件中。@TEXT函数的语法是:

```
@TEXT(['FILENAME'])
```

其中，FILENAME 是要把求解结果导出至文件的名称。如果 FILENAME 省略，求解结果的数据就会导出到标准输出设备中（一般为屏幕）。在模型的数据域中，@TEXT 函数只能出现在陈述语句的左边，实现导出数据的功能。LINGO 在模型求解结束之后才向文本文件中导出数据，并且是按照 @TEXT 函数在 LINGO 中出现的先后次序逐个完成。

以下是使用 @TEXT 函数的两个例子：

```
@TEXT('RESULTS.TXT') = X;
```

LINGO 将变量 X 的值（可能是向量）导出到 RESULTS.TXT 文件中：

```
@TEXT() = DAYS, START;
```

在这个例子中，由于语句中省略了 FILENAME 参数，因此将集合元素 DAYS 及其 START 属性值导出并显示在屏幕上。

下面通过员工安排模型具体说明 @TEXT 函数的用法。

## 8.2.4 实例——在员工安排模型中使用 @TEXT 函数

这里将再次使用员工安排模型，并用 @TEXT 函数修改该模型，将其求解结果导出到文本文件中（改动部分用黑体字表示）。模型如下：

```
SETS:
  DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
DATA:
  REQUIRED = 20 16 13 16 19 14 12;
  @TEXT('OUT.TXT') = DAYS, START;
ENDDATA
MIN = @SUM(DAYS(I):START(I));
@FOR(DAYS(J):
  @SUM(DAYS(I) | I #LE# 5:
    START(@WRAP(J - I + 1, 7)))
      >= REQUIRED(J)
);
```

模型中已经添加了一个导出数据的操作：

```
@TEXT('OUT.TXT') = DAYS, START;
```

该语句将 DAYS 集合的元素列表及其相应的 START 属性值导出到了 OUT.TXT 文件中。模型求解完毕后，LINGO 就会执行这个导出数据的操作，生成 OUT.TXT 文件：

```
MON    8.0000000
TUE    2.0000000
```

```
WED     0.0000000
THU     6.0000000
FRI     3.0000000
SAT     3.0000000
SUN     0.0000000
```

这时，就可以将数据从 OUT.TXT 文件中导入到其他应用软件。例如，如果想把数据导入到 Access 中，就可以使用 Access 中的“文件|获取外部数据|导入”菜单命令，将这些数据导入到表格中。在 Access 中定义一个名称为 Start 的表格并执行以上操作步骤，可以得到如图 8-2 所示结果。

为了将数据导出到 Excel 电子数据表中，可以在 Excel 中使用“文件|打开”命令将OUT.TXT 文件打开。打开后的文件如图 8-3 所示。

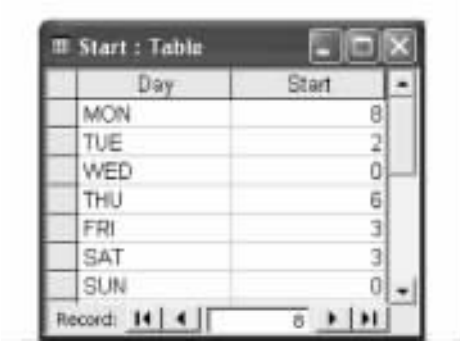



图 8-2　Access 数据表


图 8-3　Excel 数据表

在这个例子中，假设只需要输出模型目标函数的最优值和 DAYS 集合的元素列表以及 START 属性值，并且把它们同时导出到 OUT.TXT 文件中并显示在屏幕上。首先，在 LINGO 模型的数据域中添加导出数据的操作功能，使用的语句如下：

```
DATA:
  @TEXT('OUT.TXT') = DAYS, START;
  @TEXT() = DAYS, START;
ENDDATA
```

其次，还需要禁止 LINGO 求解模型后自动显示标准 LINGO 结果报告。如前所述，在 LINGO 的 Windows 版本中，选择 LINGO|Options 命令，单击 Options 对话框中的 Interface 标签，在 Interface 选项卡中选择 Terse output 复选框，然后点击 OK 按钮。在 Windows 之外的其他操作系统中，可以输入 Terse output 命令达到这个目的。现在运行模型，将得到如下简短的报告：

```
Global optimal solution found at step:        8
Objective value:                         22.00000
MON     8.0000000
TUE     2.0000000
WED     0.0000000
THU     6.0000000
```

```
FRI    3.0000000
SAT    3.0000000
SUN    0.0000000
```

## 8.3 LINGO 命令脚本

LINGO 命令脚本是包含一系列 LINGO 命令的文本文件。使用 LINGO 命令脚本时,除了需要了解 LINGO 模型的语法外,还需要知道 LINGO 的命令语句。可以把这些命令看成是一种宏语言,这些宏语言能够自动运行一些经常使用的命令以及模型。

使用"File|Take Commands"菜单命令,可以在 Windows 版本的 LINGO 中运行命令脚本;在 LINGO 的其他版本中,可以使用 TAKE 命令。在这两种情况下,都会出现一个对话框,提示需要输入包含有命令脚本的文件名。输入文件名后,LINGO 就会开始执行这个文件中的命令,直至遇到 QUIT 命令,LINGO 才会终止执行;或者当遇到文件中的终止符时,LINGO 才会恢复到正常的输入模式。

### 8.3.1 一个命令脚本实例

这里再次用员工安排模型来阐明如何使用命令脚本。假设业务已扩大,原来只有一个餐馆,而现在有 Pluto Dogs、Mars Dogs 和 Saturn Dogs 三个餐馆。这三个餐馆的员工需求如表 8-1 所示。

表 8-1 三个餐馆的员工需求

| 地点 | 周一 | 周二 | 周三 | 周四 | 周五 | 周六 | 周日 |
|---|---|---|---|---|---|---|---|
| Pluto Dogs | 20 | 16 | 13 | 16 | 19 | 14 | 12 |
| Mars Dogs | 10 | 12 | 10 | 11 | 14 | 16 | 8 |
| Saturn Dogs | 8 | 12 | 16 | 16 | 18 | 22 | 19 |

为每个餐馆运行一次模型是非常麻烦的,而且容易出错,因此可以通过构造一个自动求解三个餐馆员工安排模型的命令脚本来自动完成这个过程。构造命令脚本文件如下:

```
!允许 LINGO 把所有命令脚本反映在屏幕上
SET ECHOIN 1
!禁止 LINGO 运行模型时自动显示其标准求解报告
SET TERSEO 1
!开始一个新模型的输入
MODEL:
SETS:
  DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
```

```
DATA:
  REQUIRED = @FILE('PLUTO.LDT');
  @TEXT('PLUTO.TXT') = START;
ENDDATA
MIN = @SUM(DAYS(I):START(I));
@FOR(DAYS(J):
@SUM(DAYS(I)|I #LE# 5:
  START(@WRAP(J - I + 1, 7)))
  >= REQUIRED(J)
);
@FOR(DAYS:@GIN(START));
END
!解决 Pluto Dogs 模型
GO
!修改为 Mars Dogs 模型
ALTER ALL 'PLUTO'MARS'
!解决 Mars Dogs 模型
GO
!修改为 Saturn Dogs 模型
ALTER ALL 'MARS'SATURN'
!解决 Saturn Dogs 模型
GO
!还原参数
SET TERSEO 0
SET ECHOIN 0
```

首先,使用 SET 命令设置 LINGO 的两个参数。设置 ECHOIN 为 1,它可以让 LINGO 将所有的命令脚本显示到屏幕上,这对于调试命令脚本文件很有帮助;设置 TERSEO 为 1,它可以使 LINGO 进入概要输出模式。接着,使用了 MODEL 命令,这个命令能够使 LINGO 进入模型输入模式。LINGO 将文件中 MODEL 后面所有的文本作为模型文本并读取,直至遇到 END 命令,这个模型就成为内存里的当前模型。应该注意,在模型中起关键作用的是如下数据域:

```
DATA:
  REQUIRED = @FILE('PLUTO.LDT');
  @TEXT('PLUTO.TXT') = START;
ENDDATA
```

这里用 @FILE 函数在 LINGO 中调用外部文件中的员工需求量,同时用 @TEXT 函数将 START 的属性值写入 PLUTO.TXT 文件中。

在 END 语句后,用 GO 命令求解 Pluto Dogs 餐馆的模型。然后,用 ALTER 命令将所有关于'PLUTO'的模型改成关于'MARS'的模型。这个命令将数据域改为以下语句(黑体字表示变动部分):

```
DATA:
  REQUIRED = @FILE('MARS.LDT');
  @TEXT('MARS.TXT') = START;
ENDDATA
```

只要把 Mars Dogs 餐馆的员工需求数据放在 MARS.LDT 文件中,模型就可以再次运行,只是这一次求解出的是 Mars Dogs 餐馆的 START 属性值,同样也要求解出 Saturn Dogs 餐馆的 START 属性值。最后,用两个 SET 命令恢复刚被修改的 TERSEO 和 ECHOIN 参数。

执行该命令脚本就可以发现,LINGO 在其默认文件夹中自动生成了三个结果文件:PLUTO.TXT、MARS.TXT 和 SATURN.TXT。这三个文件分别包含三个餐馆(Pluto Dogs、Mars Dogs 和 Saturn Dogs)的 STSRT 属性的最优值。

### 8.3.2 AUTOLG.DAT 脚本文件

LINGO 中有一个选项,用户每次启动 LINGO 时都自动执行一个命令脚本。要实现这个功能,只需要将这个命令脚本命名为 AUTOLG.DAT 并且把它保存在 LINGO 的工作目录下即可。这样,当每次启动 LINGO 时,LINGO 都会自动执行这个命令脚本文件中的命令。

## 8.4 在命令行中的指定文件

当启动 Windows 版本的 LINGO 时,LINGO 将检查命令行是否存在表 8-2 中的三个命令。

表 8-2 命令与作用

| 命令 | 作用 |
|---|---|
| -tFilename | LINGO 对 Filename 文件执行 File\|Take Commands 命令。如果在 LINGO 的工作目录中存在一个 AUTOLG.DAT 文件,那么 AUTOLG.DAT 文件将优先于选择的脚本文件执行 |
| -oFilename | LINGO 对 Filename 文件执行 File\|Open 命令,并且将该文件读到一个标准窗口中 |
| -lFilename | LINGO 执行 File\|Log Output 命令,将原本输出到命令窗口的所有内容发送到该命令指定的文件 Filename 中 |

作为例子,假设有如下命令脚本:

```
!输入一个小模型
MODEL:
MAX = 20 * X + 30 * Y;X < = 50;
Y < = 60;
X + 2 * Y < = 120;
END
!简洁输出模式
SET TERSEO 1
```

```
!解决该模型
GO
!打开一个文件
DIVERT SOLU.TXT
!将求解结果写入该文件中
SOLUTION
!关闭该保存有求解结果的文件
RVRT
!退出 LINGO
QUIT
```

在这个脚本文件中,输入了一个小模型并求解该模型,然后将结果报告写入文件SOLU.TXT中。假设这个命令脚本文件名为 TEST.LTF,通过向 LINGO 的命令行中添加命令-tTEST.LTF,就能够指示 LINGO 自动执行这个 TEST.LTF 文件。


图 8-4 LINGO 快捷方式

如果在 Windows 环境下完成此功能,应该首先为 LINGO 创建一个快捷方式的图标:在桌面上单击鼠标右键,然后选择“新建”命令以及“快捷方式”命令,接着在“创建快捷方式”对话框中单击“浏览”按钮并且选择 LINGO 应用软件文件(该文件在 LINGO 主目录下,名称为 LINGO.EXE)。这样,在桌面上就会出现如图 8-4 所示的一个 LINGO 快捷方式的图标。为了编辑命令行,必须右击该 LINGO 图标,然后选择“属性”命令,这时将出现如图 8-5 所示窗口。最后,在该窗口的“Shortcut”属性页的“Target”文本框中添加-tTEST.LTF 命令。如果想让 LINGO 不打开窗口就能够运行,还需要从“Run”的下拉列表中选择“Minimized”选项。设置完毕后,单击 Apply 按钮和 OK 按钮。

这样,就可以通过双击桌面上的 LINGO 快捷方式图标运行 LINGO。以上步骤完成后,通过执行命令脚本,就可以得到下面名为 SOLU.TXT 的文本文件:

```
Variable           Value        Reduced Cost
       X        50.00000            0.000000
       Y        35.00000            0.000000
     Row    Slack or Surplus      Dual Price
       1         2050.000           1.000000
       2         0.000000           5.000000
       3         25.00000           0.000000
       4         0.000000           15.00000
```

## 8.5 重新定位输入和输出

在大多数的 Unix 操作系统中,可以将 LINGO 产生的所有屏幕输出重新定位而转到一个文本文件中。同样,也可以将键盘的输入全部重新定位到一个输入文件中。在指定的文件中使用如下语句就可以完成重新定位的功能:


**图 8-5　属性对话框**

```
LINGO < input_file > output_file
```

**当执行这个命令后，**LINGO **将从** input _ file **中读出内容并且将自动生成一个名为** output _ file **的文本文件。该文件包含** LINGO **正常情况下传送到屏幕上的所有内容，文件路径包含在输入和输出的文件名中。**

**使用这个功能可以在大型应用软件中将** LINGO **作为“黑匣子”。如果操作正确，用户将不会发现** LINGO **正在后台运行。**

**例如，通过以下命令来改变输入和输出，运行** TEST. LTF **命令脚本文件可以生成一个与** SOLU. TXT **同样的求解结果文件** CAPTURE. TXT**：**

```
LINGO < TEST.LTF > CAPTURE.TXT
```

**这时，**CAPTURE. TXT **文件用于捕捉所有的屏幕输出。**

# 第 9 章　与电子数据表的接口

如上一章所提到的,如果将数据保存在 LINGO 模型文件中非常麻烦并且效率很低。实际上,大多数模型中都会用到大量的数据。电子数据表比较适合管理小规模和中等规模的数据,同时,它还能够方便地操作和显示模型所产生的结果。基于这些原因,LINGO 提供了很多允许用户从电子数据表中导入数据以及将求解结果导入电子数据表的功能。LINGO 的这些功能包括连接到 Excel 的实时对象连接和嵌入(OLE)连接、OLE 自动连接功能(用于从 Excel 的宏里驱动 LINGO)、嵌入的 OLE 连接(用于将 LINGO 的功能引入到 Excel 中)。目前,只有 Windows 版本的 LINGO 才具有以上功能。

## 9.1　从电子数据表中导入数据

LINGO 提供的@OLE 函数用于从电子数据表中导入数据。该函数能够直接完成 LINGO 和 Excel 之间的数据传输。

### 9.1.1　使用@OLE 函数从 Excel 中导入数据

@OLE 函数是一个接口函数,它使用 OLE 传输完成从 Excel 中输入和输出数据的任务。OLE 传输直接在内存中进行,不使用任何中间文件。当使用@OLE 函数时,LINGO 装载 Excel,同时告诉 Excel 装载需要的电子数据表,并且请求电子数据表中相应域中的数据。但是,必须有 Excel 5.0 或更新版本才能够使用@OLE 函数。@OLE 函数只能用在模型的数据域和初始化域中。

@OLE 函数既可以读取数据集合元素,也可以读取集合的属性值。其中,集合元素必须以文本格式读取,而属性值则必须以数字格式读取。基本集合的每一个元素需要一个单元格的数据,而为了初始化每一个 n 维衍生集合成员,都需要 n 个单元格的值,即前 n 个单元格包含第一个集合的元素,接着的 n 个单元格包含第二个集合的元素,等等。

@OLE 函数可以读取一维或二维数据域(即存放于 Excel 文件的同一个工作表中的数据域),但是它不能读取不连续的或三维的数据域(即分布在 Excel 文件的不同工作表中的数据域),并且从左向右、从上到下依次读取数据域。

LINGO 模型中,在数据域和初始化域中使用@OLE 函数导入数据的语法如下:

```
object_list = @OLE('spreadsheet_file'[, range_name_list]);
```

其中,object_list 是一个模型对象的列表,它可以由逗号分隔,object_list 可以包含集合名称、属性、数量变量的任意组合;spreadsheet_file 是 LINGO 读取数据的 Excel 电子数据表的文件名;range_name_list 是 LINGO 从中读取数据的电子数据表中域名的列表。对于 object_list 中的每一个元素,都要有一个域名与之对应,并且只能有一个域名与之对应。

指定电子数据表的域一般有以下三种方法。第一,完全省略域名参数。在这种情况下,LINGO 将 object _ list 中所列出的对象名作为域的名称。第二,可以指定一个域名来获得所有的数据。在这种情况下,object _ list 中的所有对象必须定义在同一个集合中,这样 LINGO 就好像是从数据表中读取数据。当需要为多个对象指定单个域名时,所有的对象必须是相同的数据类型。此外,不能将集合元素(文本)和集合属性值(数字)混在一起。第三,针对 object _ list 中的每一个对象分别指定一个域名。在这种情况下,这些对象不需要在同一个集合中定义,也不需要是相同的数据类型。利用这三种方法使用 @OLE 函数的例子分别如下:

```
COST, CAPACITY = @OLE('SPECS.XLS');
```

在这个例子中,@OLE 函数并没有指定域名。因此,LINGO 默认使用模型对象的名称,即在 Excel 文件 SPECS. XLS 中,名为 COST 和 CAPACITY 的域中的数据值将初始化 COST 和 CAPACITY 对象。

```
COST, CAPACITY = @OLE('SPECS.XLS', 'DATATABLE');
```

在这个例子中,只指定了一个域用于初始化 COST 和 CAPACITY。假设 DATATABLE 域包含两列数据,LINGO 将用第一列的值初始化 COST 对象,用第二列的值初始化 CAPACITY 对象。注意:COST 和 CAPACITY 必须定义在同一个集合中,并且它们要么都是集合元素,要么都是集合属性,即它们必须是相同的数据类型。

```
COST, CAPACITY = @OLE('SPECS.XLS', 'COST01', 'CAP01');
```

在这个例子中,为了初始化 COST 和 CAPACITY,分别为它们指定了一个域名。LINGO 用 COST01 域中的值初始化 COST 对象,用 CAP01 域中的值初始化 CAPACITY 对象。

## 9.1.2 在运输模型中使用 @OLE 函数导入数据

下面利用 Wireless Widgets 公司的模型实例进一步说明在 LINGO 中怎样使用 @OLE 函数。该模型被改动的部分用黑体字表示:

```
SETS:
  WAREHOUSES: CAPACITY;
  VENDORS: DEMAND;
  LINKS(WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
  MIN = @SUM(LINKS(I, J):
  COST(I, J) * VOLUME(I, J));
  @FOR(VENDORS(J):
  @SUM(WAREHOUSES(I):
  VOLUME(I, J)) = DEMAND(J));
  @FOR(WAREHOUSES(I):
  @SUM(VENDORS(J): VOLUME(I, J))
  <= CAPACITY(I));
DATA:
```

```
    WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
    @ OLE ('C :\ LINGO \ SAMPLES \ WIDGETS.XLS',
    'WAREHOUSES', 'VENDORS', 'CAPACITY',
    'DEMAND', 'COST') ;
ENDDATA
```

在这个模型中并没有明确给出数据,即没有在数据域中输入数据,而是将数据放入 WIDGETS.XLS 电子数据表中。图 9-1 所示是 WIDGETS.XLS 文件。

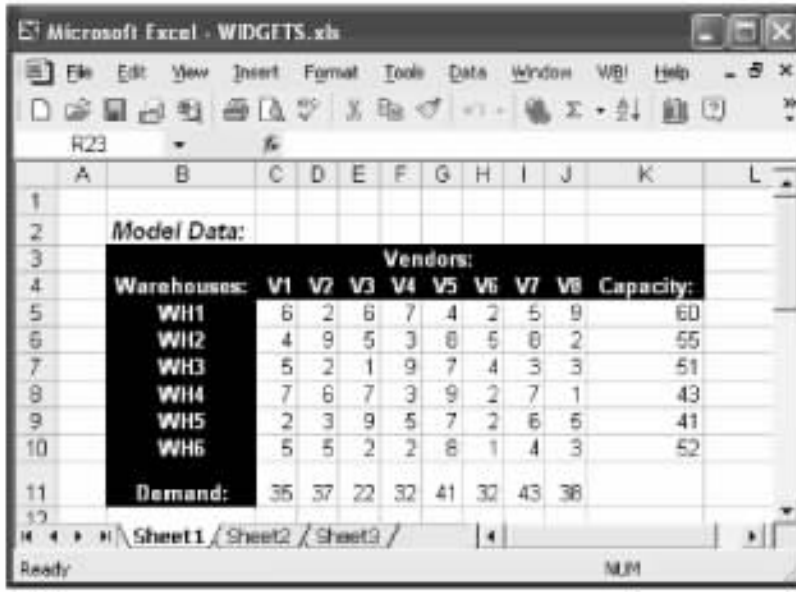



图 9-1　WIDGETS.XLS 文件

表 9-1　域名及范围

| 域名 | 范围 |
|---|---|
| Capacity | K5 :K10 |
| Cost | C5 :J10 |
| Demand | C11 :J11 |
| Vendors | C4 :J4 |
| Warehouses | B5 :B10 |

除了向这个电子数据表中输入数据,还需要为 COST、CAPACITY、DEMAND、VENDOR 和 WAREHOUSE 指定相应的域名。指定的域名如表 9-1 所示。

在 Excel 中,可按如下方法定义域名:

1) 按下鼠标左键并拖曳鼠标选择数据区域;

2) 释放鼠标;

3) 选择 Insert | Name | Define 命令;

4) 输入正确的域名;

5) 单击 OK 按钮。

在该模型的数据域中,使用以下的 @OLE 函数将数据从 Excel 导入到 LINGO 中:

```
    WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
      @ OLE ('C :\ LINGO \ SAMPLES \ WIDGETS.XLS',
      'WAREHOUSES', 'VENDORS', 'CAPACITY',
      'DEMAND', 'COST') ;
```

**注意** 由于该模型中的对象都是基本集合元素及属性，并且它们的名称和 Excel 中对应的域名相同，因此，可以省略 @OLE 函数中的域名列表参数，即使用如下更简单的命令语句：

```
WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
  @OLE('C:\LINGO\SAMPLES\WIDGETS.XLS');
```

这里只用一个 @OLE 函数为该模型获取所有的数据。这并不是必需的，还可以使用多个 @OLE 函数，比如对于模型中的每个对象都使用一个 @OLE 函数。

当开始求解这个模型时，LINGO 将首先运行 Excel（假设它还没有运行）并且装载 WIDGETS 电子数据表，然后利用电子数据表中的数据初始化 CAPACITY、COST、DEMAND 属性以及 WAREHOUSES 和 VENDORS 集合。成功运行后 LINGO 将出现如下结果报告：

```
Optimal solution found at step:       16
Objective value:                    664.0000
  Variable           Value        Reduced Cost
VOLUME(WH1, V2)    19.00000      0.0000000
VOLUME(WH1, V5)    41.00000      0.0000000
VOLUME(WH2, V4)    32.00000      0.0000000
VOLUME(WH2, V8)    1.000000      0.0000000
VOLUME(WH3, V2)    12.00000      0.0000000
VOLUME(WH3, V3)    22.00000      0.0000000
VOLUME(WH3, V7)    17.00000      0.0000000
VOLUME(WH4, V6)    6.000000      0.0000000
VOLUME(WH4, V8)    37.00000      0.0000000
VOLUME(WH5, V1)    35.00000      0.0000000
VOLUME(WH5, V2)    6.000000      0.0000000
VOLUME(WH6, V6)    26.00000      0.0000000
VOLUME(WH6, V7)    26.00000      0.0000000
```

## 9.2 将求解结果导出到电子数据表中

### 9.2.1 使用 @OLE 函数将求解结果导出到 Excel 中

上一节已经讲述了如何从电子数据表中导入数据，本节介绍如何用 @OLE 函数将求解结果导出到电子数据表中。用 @OLE 函数导出数据的语法是：

```
@OLE('spreadsheet_file'[, range_name_list]) = object_list;
```

其中，object_list、spreadsheet_file 和 range_name_list 的含义同上一节所述。

考虑以下模型以及它的求解结果：

```
SETS:
  S1:X;
```

```
  S2(S1, S1): Y;
ENDSETS
DATA:
  S1,X = M1,1  M2,2  M3,3;
  S2,Y = M1,M2,4  M3,M1,5;
ENDDATA
  Variable           Value
      X(M1)        1.000000
      X(M2)        2.000000
      X(M3)        3.000000
    Y(M1, M2)      4.000000
    Y(M3, M1)      5.000000
```

X 和 Y 都是集合属性,对于每一个元素都导出一个数值。也就是说,X 导出 1、2、3,而 Y 导出 4 和 5。S1 是一个基本集合,它对于每一个元素都导出一个文本值,即 M1、M2 和 M3。S2 是一个二维衍生集合,对于每一个元素它都导出两个文本值。在本例中,S2 有两个成员,因此它导出四个值 M1、M2、M3、M1。

和使用 @OLE 函数导入数据一样,同样有三种方法指定域名。利用这三种方法指定域名的例子如下:

```
@OLE('C:\XLS\DEVELOP.XLS','BUILD_IT','HOW_BIG') = BUILD, SQ_FEET;
```

在本例中,对于每一个模型对象都定义了一个域名,因此,BUILD 的值将赋给 BUILD_IT 域;同时,SQ_FEET 的值将赋给 HOW_BIG 域。语句如下:

```
@OLE('C:\XLS\DEVELOP.XLS') = BUILD, SQ_FEET;
```

在这个例子中省略了域名参数。因此,LINGO 默认使用模型对象的名称作为域的名称,即 LINGO 将 BUILD 和 SQ_FEET 的值导出到 Excel 电子数据表 DEVELOP.XLS 中具有相同名称的域中。语句如下:

```
@OLE('C:\XLS\DEVELOP.XLS','SOLUTION') = BUILD, SQ_FEET;
```

在上例中,仅仅定义了一个域——SOLUTION 来获取两个模型对象。假设该域包含两列并且模型对象都是一维的,那么 BUILD 的值将被赋给该域的第一列,而 SQ_FEET 的值将被赋给第二列。注意:BUILD 和 SQ_FEET 必须是定义在同一个集合上的模型对象。

需要注意的是,利用 @OLE 函数作为数据导出和数据导入的最大差别是 @OLE 函数在语句中出现的位置。当 @OLE 函数出现在等号的左侧时,表示从 LINGO 中导出数据;当 @OLE 函数出现在等号的右侧时,表示向 LINGO 导入数据。因此应该分清下面格式:

```
@OLE(...) = object_list;? 导出数据
object_list = @OLE(...);? 导入数据
```

## 9.2.2 在运输模型中使用 @OLE 函数导出数据

在前面的章节中,通过 Wireless Widgets 运输模型讲解了怎样用 @OLE 函数将数据导入 Ex-

cel 中,但是并没有讲如何用 @OLE 函数将 LINGO 的求解结果导入电子数据表中。现在,为了能够将求解结果导入电子数据表中,需要对模型作适当的改动。该模型数据域被改动的部分用黑体字表示:

```
SETS:
  WAREHOUSES:CAPACITY;
  VENDORS:DEMAND;
  LINKS(WAREHOUSES, VENDORS):COST, VOLUME;
ENDSETS
  MIN = @SUM(LINKS(I, J):
  COST(I, J) * VOLUME(I, J));
  @FOR(VENDORS(J):
  @SUM(WAREHOUSES(I):
  VOLUME(I, J)) = DEMAND(J));
  @FOR(WAREHOUSES(I):
  @SUM(VENDORS(J):VOLUME(I, J))
  <= CAPACITY(I));
DATA:
  WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
  @OLE('C:\LINGO\SAMPLES\WIDGETS.XLS',
  'WAREHOUSES', 'VENDORS', 'CAPACITY',
  'DEMAND', 'COST');
  @OLE('C:\LINGO\SAMPLES\WIDGETS.XLS',
  'VOLUME') = VOLUME;
ENDDATA
```

本模型用 @OLE 函数将模型中含有决策变量的属性 VOLUME 导出到 Excel 的 WIDGETS.XLS文件中。由于属性名称和域名相同,因此可以在 @OLE 函数中省略数据域名称。用下面的简单语句

```
@OLE('\LINGO\SAMPLES\WIDGETS.XLS') = VOLUME;
```

表示时,需要在 WIDGETS 电子数据表中添加名称为 VOLUME 的域来接受从 LINGO 中导出的求解结果,方法参见上一节的内容。

求解这个模型后,LINGO 就会将 VOLUME 属性值导出到电子数据表中,更新后的 VOLUME 域如图 9-2 所示。

### 9.2.3 输出统计报告

只要用 @OLE 函数将求解结果导出到电子数据表中,都将得到一份关于导出过程的摘要,这个摘要称为输出统计报告 (Export Summary Report)。这个报告将出现在求解结果报告窗口的顶端。针对模型中用于导出求解结果的每一个 @OLE 函数,LINGO 都会生成一个输出统计报告。以下是在 Wireless Widgets 运输模型中使用 @OLE 函数将求解结果导出到 Excel 时得到的

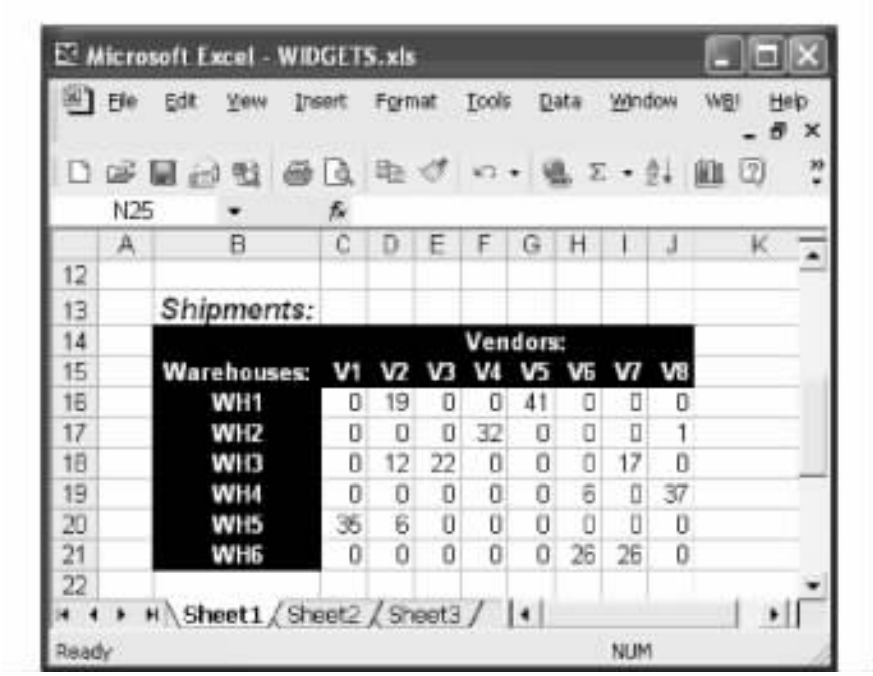



图 9-2　VOLUME 域

**一份输出统计报告：**

```
Export Summary Report
- - - - - - - - - - - - - - - - - - - - - - -
Transfer Method:  OLE BASED
Spreadsheet:       \ LINGO \ SAMPLES \ WIDGETS.XLS
Ranges Specified:              1
Ranges Found:                  1
VOLUME
Range Size Mismatches:         0
Values Transferred:           48
```

**传输方法**（Transfer Method）**部分列出了使用** @OLE **函数的传输类型，本例中** @OLE **传输用“**OLE BASED**”表示。**

**电子数据表**（Spreadsheet）**部分列出了目标电子数据表的名称**（**包括路径**）**。**

**指定域**（Ranges Specified）**部分列出了在导出函数中指定的域的总个数。**

**被发现的域**（Ranges Found）**部分列出了电子数据表中实际被发现的域**（**和导出函数中指定的域相对应**）**的个数。**

**一般情况下，对于需要导出到** Excel **中的每一个数据值，电子数据表的域都应该有一个单元格与之对应。如果某个域中的单元格过多或过少，都将导致域的大小不匹配。缺少的或多余的单元格的个数将在域规模不匹配**（Range Size Mismatches）**部分中列出。**

**数据传输**（Values Transferred）**部分列出了实际传输到所有指定域中的数据的总个数。**

## 9.3 利用 OLE 技术实现与 Excel 的自动连接

LINGO 的命令脚本可以放在 Excel 的域中并通过 OLE 技术自动将该脚本传递给 LINGO，这样就建立了 Excel 和 LINGO 之间的客户端-服务器（Client-Server）关系。

为了能够阐述清楚这个特征，下面再次使用员工安排模型。这个例子需要读者对 Excel 中的 Visual Basic 宏的使用比较熟悉。

考虑图 9-3 所示的 Excel 数据表。

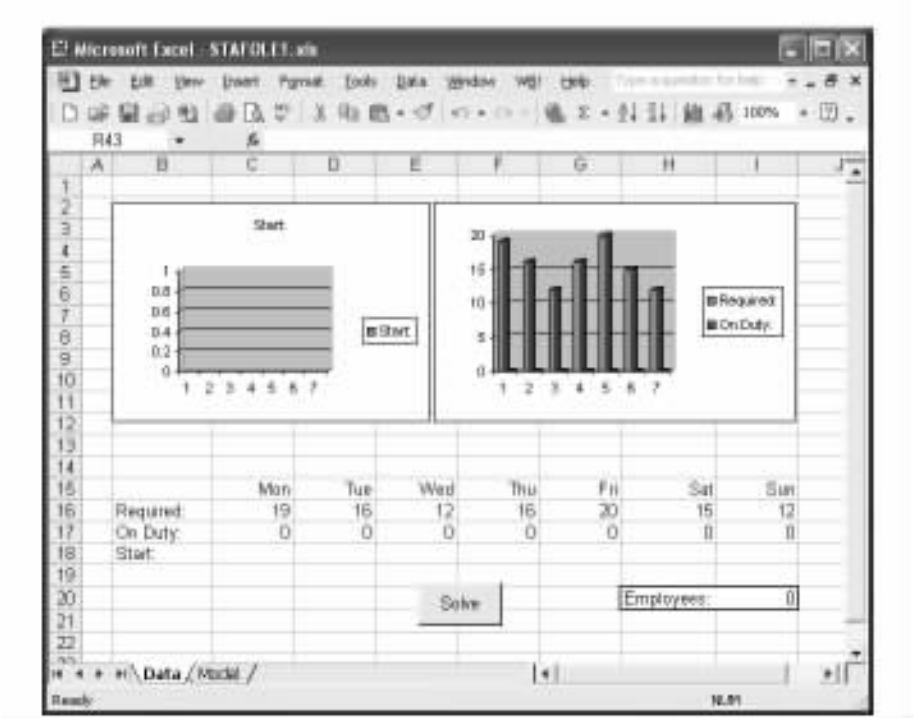



图 9-3 求解前的 STAFOLE1.XLS|Data 页

在该数据表中，已经将员工需求量放置于 C16:I16 域中，并且将该域命名为 REQUIREMENTS。同时，指定 C18:I18 域的名称为 START，LINGO 将会把求解结果导出到 START 域中。在该数据表中，通过引入两个图表可以使得结果更加形象，有助于帮助分析。左边的图表表示一周内的每一天需要有多少员工从这一天开始工作，右边的图表用于比较每一天需要的员工数和实际工作的员工数。

该数据表中还有一个名为 Model 的工作表。选中这一页，将会看到图 9-4 所示内容。

可见，STAFOLE1.XLS 中的第二页包含了员工安排模型的命令脚本。第 1 行命令 SET ECHOIN 1 用于打开终端响应功能。当 LINGO 读到这个命令后就会将该命令脚本读到命令窗口中。从第 2 行到第 21 行是模型的内容。

注意 在模型的 Data 部分使用了两个 @OLE 函数。第一个用于将数据从电子数据表导入，第二个用于将求解结果导出到电子数据表中。也就是说，数据从数据表的第一个的

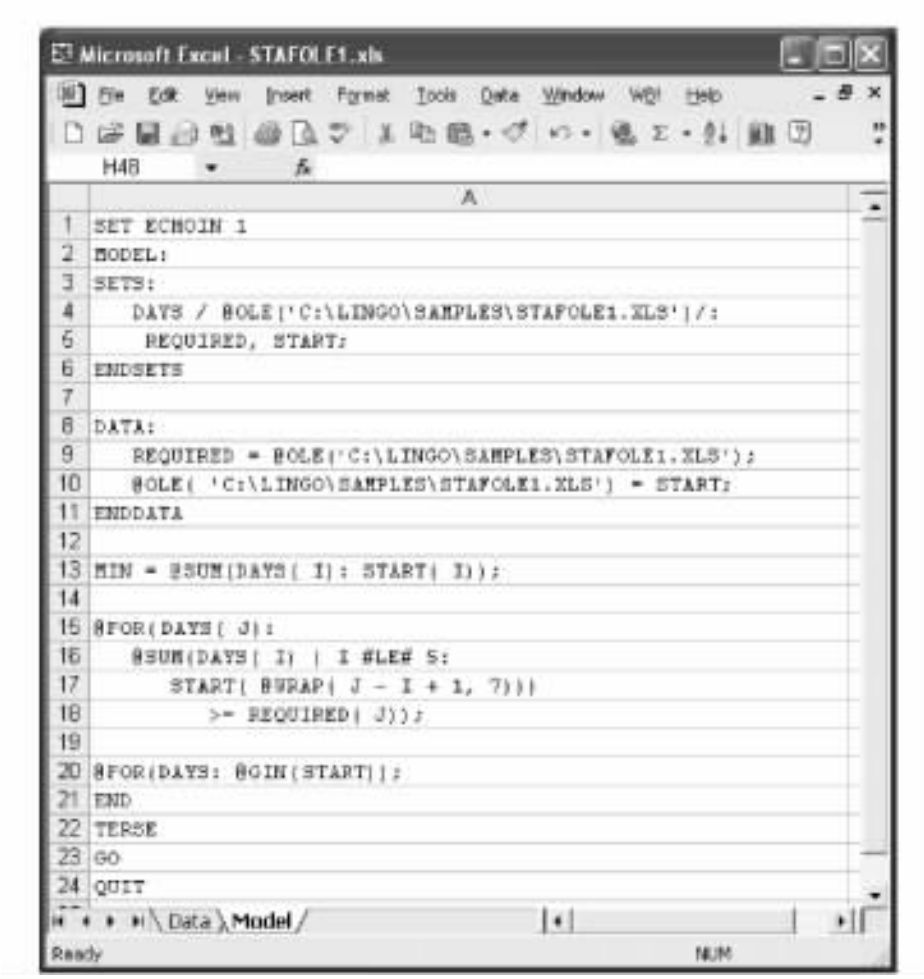



图 9-4 STAFOLE1.XLS|Model 页

REQUIRED 域中读出并且将求解结果写入第一页的 START 域中。在第 22 行,使用 GO 命令求解该模型。另外,还需要指定包含这个脚本的 A1:A23 域的名称为 MODEL。

假设已经在电子数据表中完成了 LINGO 命令脚本,那么接下来的问题是怎样将它送入 LINGO 并运行。这里需要用到 OLE 自动技术。在 Data 页中,有一个名为 Solve 的按钮。将这个按钮添加到该电子数据表中,并且附上如下的 Excel Visual Basic 宏语句:

```
Sub LINGOSolve()
  Dim iErr As Integer
  iErr = LINGO.RunScriptRange("MODEL")
  If (iErr > 0) Then
    MsgBox ("Unable to solve model")
  End If
End Sub
```

在上述语句中,使用了通过 OLE 自动技术调用 LINGO 的 RunScriptRange 方法,并且将

MODEL 域传递给它。RunScriptRange 方法驱使 Excel 获取 MODEL 域的内容并开始处理其中的命令。只有当遇到 QUIT 命令或域中不再包含任何命令时,这个处理过程才会停止。

如果成功处理了脚本,RunScriptRange 命令的返回值为 0 如果处理不成功,它将返回如表 9-2 所示的错误代码。

表 9-2 RunScriptRange 返回的错误代码

| 错误代码 | 意义 | 错误代码 | 意义 |
|---|---|---|---|
| 1 | 非法语句 | 9 | 无法分配内存 |
| 2 | 保留的 | 10 | 无法配置脚本阅读器 |
| 3 | 无法打开日志文件 | 11 | LINGO 忙 |
| 4 | 无效的脚本 | 12 | OLE 异常 |
| 5 | 非法数列格式 | 13 | 无法初始化 Excel |
| 6 | 非法数列维数 | 14 | 无法读取 Excel 域 |
| 7 | 非法数列边界 | 15 | 无法找到 Excel 域 |
| 8 | 无法锁住数据 | | |

另外,还需要在该电子数据表的 Data 页中添加如下的 Auto _ Open () 子函数:

```
Dim LINGO As Object
Sub Auto _ Open ()
  Set LINGO = CreateObject ("LINGO.Document.4")
End Sub
```

利用上述语句,当此电子数据表每一次被打开时,上述宏都会自动执行。该宏将 LINGO 声明为一个对象,并且用 CreateObject 函数将该对象与 LINGO 软件对应起来。

这时,返回到电子数据表的第一页并且单击 Solve 求解按钮,可以看到如图 9-5 所示的最优解。

可见,对于一周中的每一天,从该天开始工作的最佳员工数就显示在 START (C18 :I18) 域中。同时,用于反映结果的两个图表也按照这个求解结果有了一定的更新。

## 9.4 将 LINGO 模型嵌入 Excel 中

除了上文讲述的方式外,LINGO 还可以作为一个 OLE 服务器使用。也就是说,可以将 LINGO 模型嵌入到任何一个可以作为 OLE 容器使用的应用软件中,而 Excel 就是这样的一种应用软件。将 LINGO 模型嵌入 Excel 后,只要打开这个电子数据表,就可以立即获得 LINGO 模型,也就不需要再考虑何时启动 LINGO 以及寻找与本数据表相对应的 LINGO 模型等工作。

为了将一个 LINGO 模型嵌入到一个电子数据表中,可以选择 Excel 中的菜单命令 Insert | Object (插入 | 对象),这样将看到一个可嵌入对象的列表。选择图 9-6 所示列表中的 LINGO Document 对象,单击 OK 按钮后,一个空白的 LINGO 模型窗口将被嵌入到电子数据表中。这时,可以像在 LINGO 中那样在这个模型窗口中直接输入文本,或者从其他应用程序中复制一

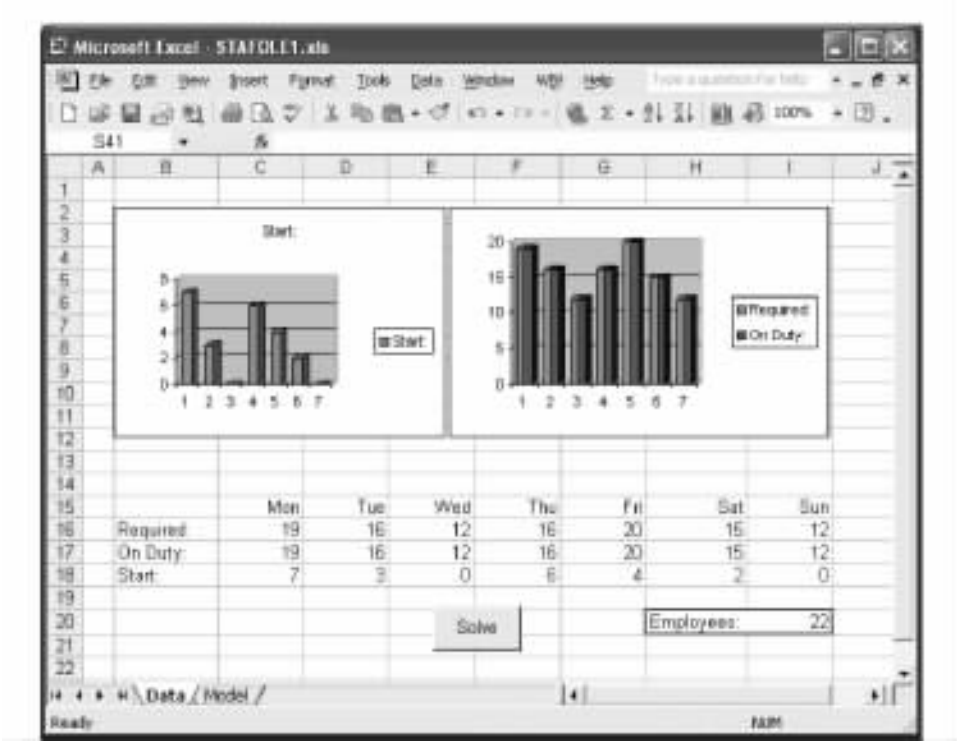



图 9-5　求解后的 STAFOLE1.XLS | Data 页

些内容到该窗口中。当保存电子数据表时，嵌入的 LINGO 模型也将自动和它一起保存。同样，当在 Excel 中打开该电子数据表时，嵌入的 LINGO 模型也将自动调入。

为了能够阐述清楚这个特征，再次使用员工安排模型的例子。在这个例子中，电子数据表将包含模型所需的数据以及嵌入的 LINGO 模型，并且求解结果最终将由 LINGO 导出到该电子数据表中。这个例子可以在 LINGO h SAMPLES 文件夹中找到，名称为 STAFOLE2.XLS。如果在 Excel 中打开这个电子数据表，可以看到图 9-7 所示内容。

和前面的例子一样，分别定义 REQUIRED 域和 START 域。在该电子数据表的右上角，定义了一个用于将求解结果更加形象化的图表。电子数据表的左上角是一个标注为 < Embedded LINGO Model > 的区域，这个区域包含了一个员工安排模型。双击这个区域，可以看到图 9-8 所示模型。

当激活这个 LINGO 模型后，LINGO 的菜单和工具栏将取代 Excel 的菜单和工具栏。因此，当在 Excel 中使用嵌入式 LINGO 模型时，可以获得 LINGO 的所有功能。当取消选定 LINGO 模型时，Excel 的菜单和工具栏将会自动出现。嵌入 OLE 的强大功能就在于它允许用户将两个或更多应用软件的功能结合起来，就好像它们是一个单独、完整的应用软件一样。

通过拖曳 LINGO 模型区域的右下角，可以看到模型的完整内容：

```
SETS:
  DAYS / MON TUE WED THU FRI SAT SUN/:
  REQUIRED, START;
ENDSETS
```

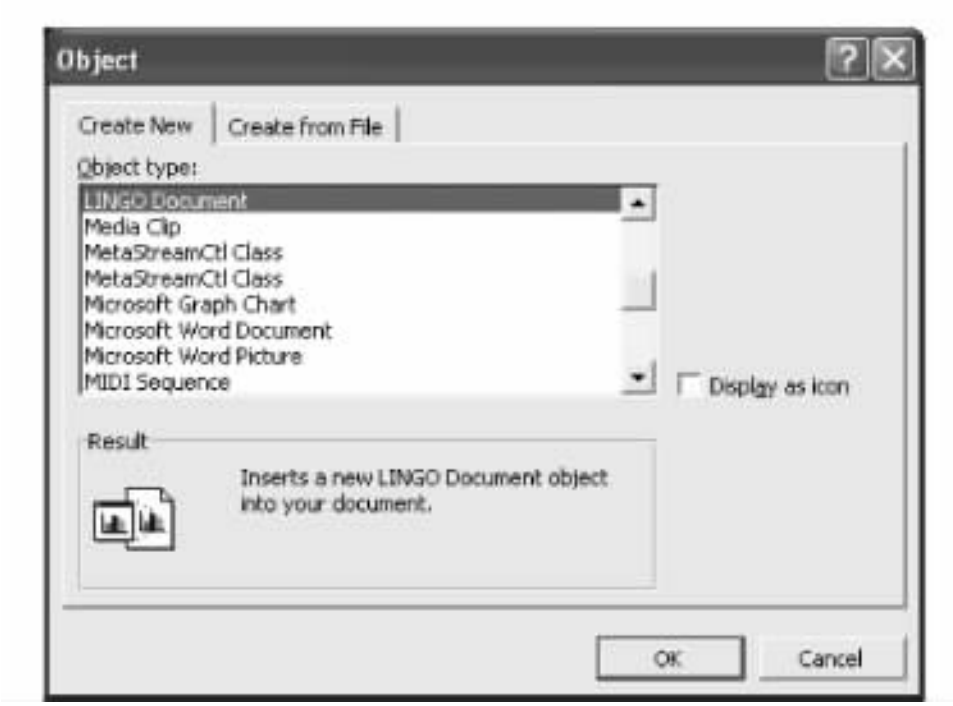



**图 9-6　插入对象对话框**

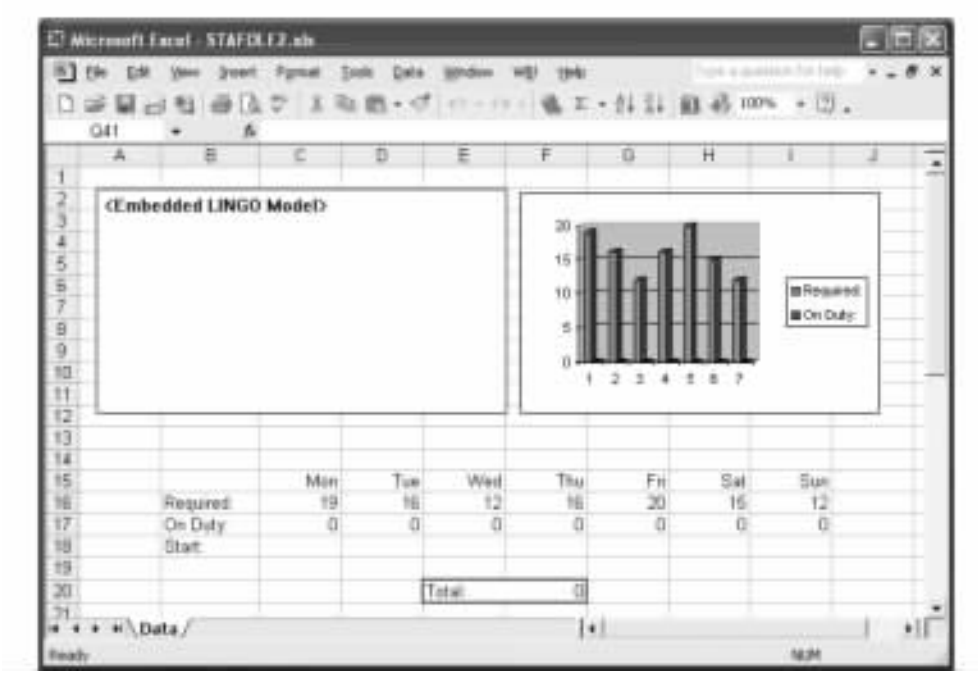



**图 9-7　STAFOLE2.XLS|Data 页**

```
DATA:
  REQUIRED =
  @OLE('C:\LINGO\SAMPLES\STAFOLE2.XLS');
```

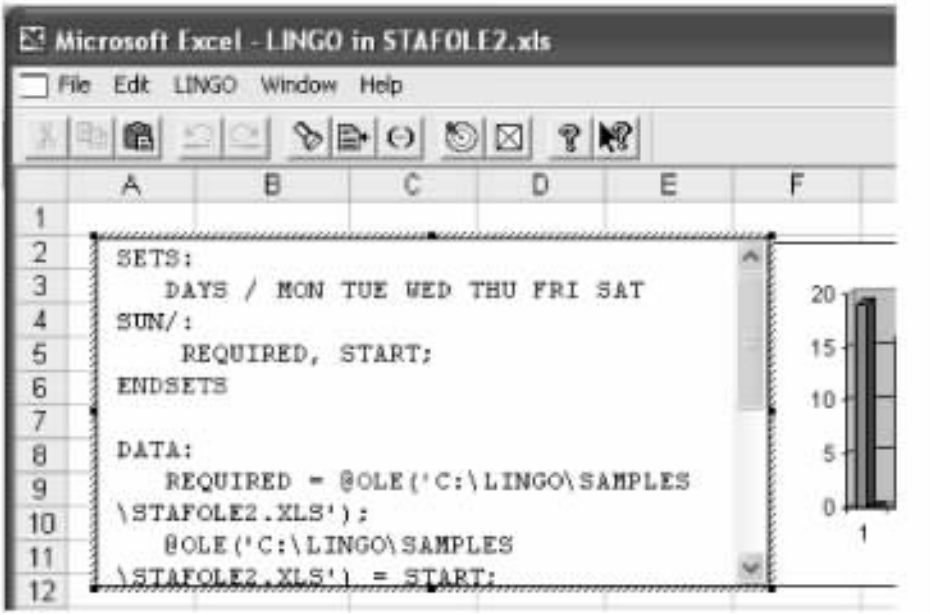



**图 9-8　STAFOLE2.XLS|员工安排模型**

```
  @OLE('C:\LINGO\SAMPLES\STAFOLE2.XLS')
  = START;
ENDDATA
MIN = @SUM(DAYS:START);
@FOR(DAYS(J):
  @SUM(DAYS(I)|I #LE# 5:
    START(@WRAP(J - I + 1, 7)))
      >= REQUIRED(J));
@FOR(DAYS:@GIN(START));
```

**如果双击包含 LINGO 模型的区域，LINGO 菜单将出现在屏幕的顶部，选择 LINGO|Solve 菜单命令，就可以运行求解该模型。当 LINGO 完成模型的最优化后，它将求解结果送回到电子数据表中，运行结果如图 9-9 所示。**

## 9.5　在 LINGO 模型中嵌入 Excel 表格

**与嵌入 LINGO 模型到电子数据表中一样，还可以反过来将电子数据表嵌入到 LINGO 模型中。为了说明这一点，打开文件夹 LINGO 中的子目录 SAMPLES 下的 STAFFOLE 模型，从而可以得到如图 9-10 所示的员工安排模型。**

**这个模型要从 Excel 文件 STAFFOLE.XLS 中读取数据并且把求解结果送回。为了更容易一些，最好把该电子数据表嵌入到 LINGO 模型里，这样就可以避免每次使用模型时都要装载它。为此，在 LINGO 中选择 Edit|Insert New Object 菜单命令，将得到如图 9-11 所示的对话框。**

**单击 Create from File (从文件创建) 单选按钮，在其中的文本框中输入电子数据表文件名，**

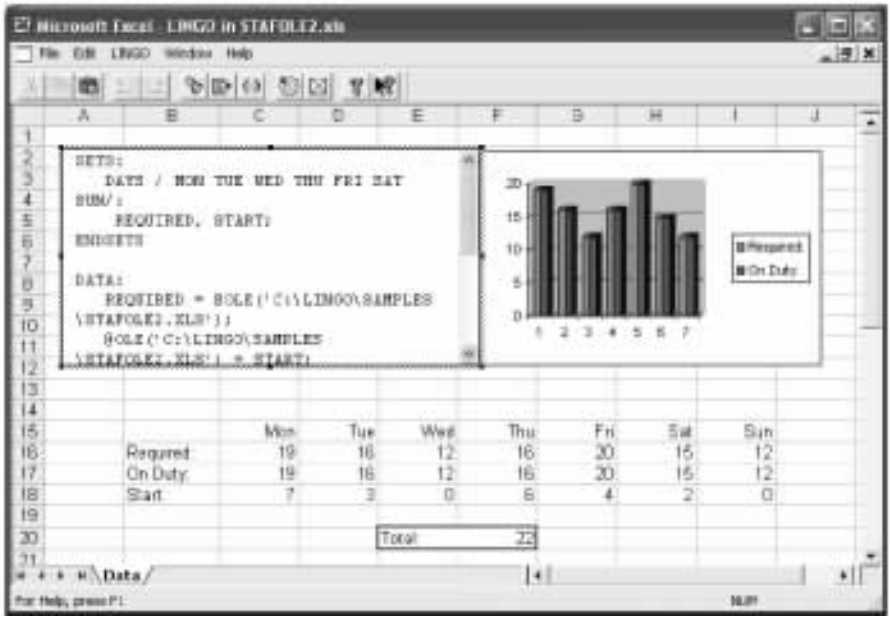



**图 9-9　STAFOLE2.XLS|求解结果**

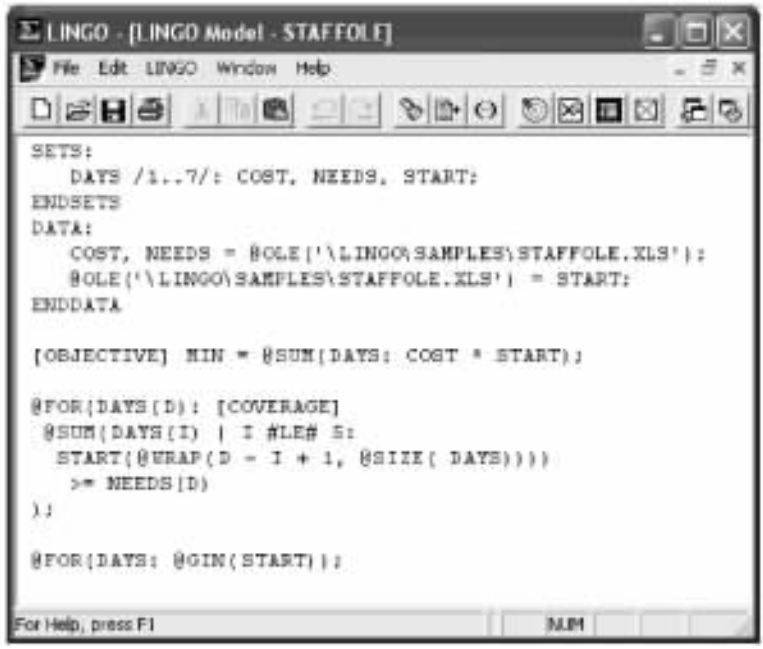



**图 9-10　STAFFOLE 模型**

**单击 Link 复选框，然后点击 OK 按钮，可以得到如图 9-12 所示的 LINGO 模型。**

**这时，STAFFOLE.XLS 已经被嵌入到 LINGO 模型的顶部。只要双击该电子数据表，就可以对它进行编辑。当保存 LINGO 模型时，连接的 Excel 文件也将被保存。求解该模型后，将会得到图 9-13 所示的结果。**

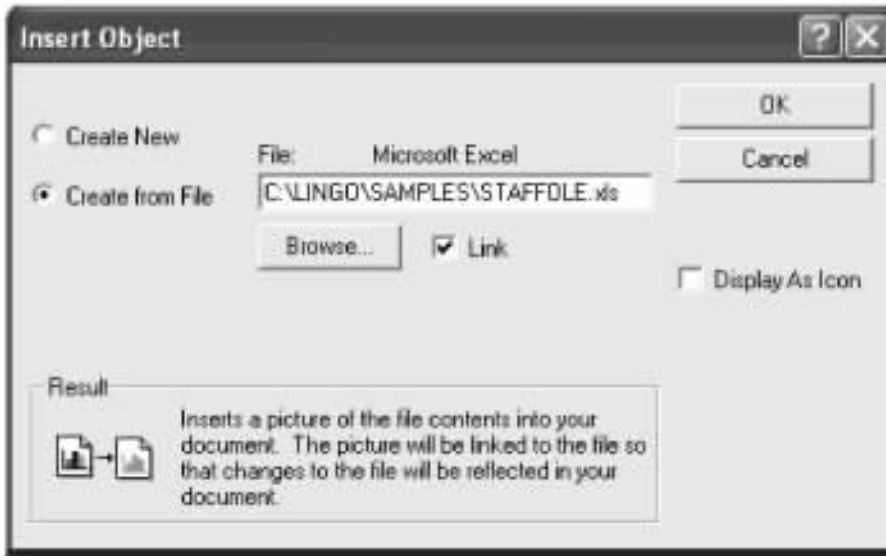



**图 9-11　插入对象对话框**

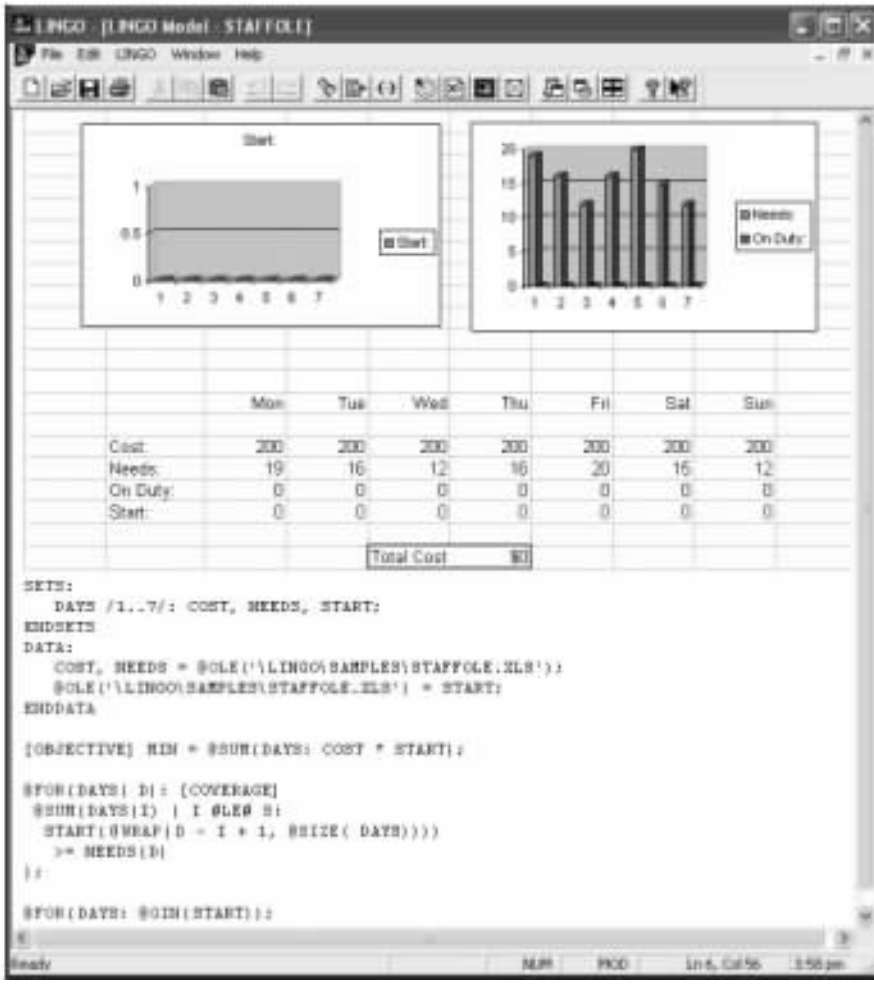



**图 9-12　嵌入 Excel 表的 STAFFOLE 模型**

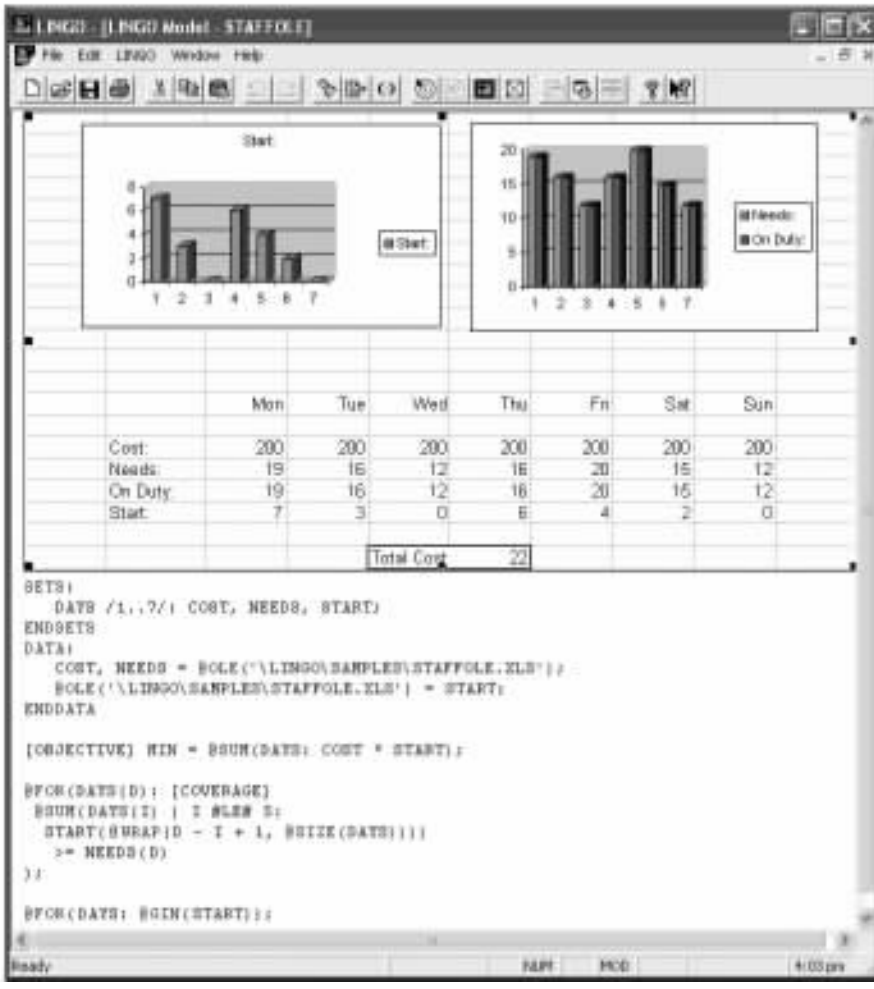



**图 9-13　STAFFOLE 模型的求解结果**

# 第 10 章　与数据库的接口

电子数据表软件适用于管理小规模和中等规模的数据。如果模型需要处理大规模的数据,数据库管理系统(DBMS)无疑是最好的工具。在许多商业建模问题中可以发现,大多数的数据都保存在一个或多个数据库中。基于这些原因,LINGO 支持含有开放式数据库连接(Open Database Connectivity,ODBC)驱动程序的任何 DBMS 的连接。ODBC 为 DBMS 定义了一个标准化接口,通过这个标准化接口,LINGO 可以访问任何 ODBC 支持的数据库。具体地讲,LINGO 为 Access、DBase、Excel、FoxPro、Oracle、Paradox、SQL Server 和 Text Files 安装了 ODBC 驱动程序。

如果要使用上面没有列出来的数据库,则需要安装相应的数据库驱动程序。

为了访问数据库,LINGO 提供了一个名为@ODBC 的接口函数。@ODBC 函数可以从任何 ODBC 数据源中导出数据或将数据导入到任何 ODBC 数据源中。但是,@ODBC 函数只能用于 Windows 版本的 LINGO 中。

## 10.1　ODBC 数据源

当安装 Windows 版本的 LINGO 时,也同时安装了一个小型的运输模型。该模型和前面章节所讲述的运输模型有一点不同,即它的所有数据都是从数据库中得到的,并且将模型的求解结果写入到同一个数据库中。这个模型文件的名称是 TRANDB.LG4,保存在 h LINGO h SAMPLES h 文件夹中。该模型文件的内容如下:

```
MODEL:
  !一个 3 个 PLANT (工厂)、4 个 CUSTOMER (消费者) 的运输问题;
  !数据可以通过 ODBC 连接从 Access 数据库或 Oracle 数据库中得到。必须首先使用 ODBC 管理器以"Transportation"为数据源名注册一个已有的数据库,这样模型就可以运行了。更多的细节请见下文:
TITLE Transportation;
SETS:
  PLANTS:CAPACITY;
  CUSTOMERS:DEMAND;
  ARCS(PLANTS, CUSTOMERS):COST, VOLUME;
ENDSETS
!目标函数;
  [OBJ] MIN = @SUM(ARCS:COST * VOLUME);
!需求约束;
  @FOR(CUSTOMERS(C):
    @SUM(PLANTS(P):VOLUME(P, C)) >= DEMAND(C));
```

```
    !供给约束;
      @FOR(PLANTS(P):
        @SUM(CUSTOMERS(C):VOLUME(P,C))<=CAPACITY(P));
    DATA:
      !通过ODBC导入数据;
      PLANTS,CAPACITY     =@ODBC();
      CUSTOMERS,DEMAND    =@ODBC();
      ARCS,COST           =@ODBC();
      !通过ODBC导出求解结果;
      @ODBC()    =VOLUME;
    ENDDATA
  END
```

可见,在该模型的数据域中,通过使用@ODBC函数建立了与ODBC数据源的连接,以读取所有数据并输出最终求解结果。

ODBC数据源是一个具有如下特征的数据库:

1)存放于DBMS中,这个DBMS拥有一个ODBC驱动程序;

2)必须在ODBC管理器中注册,如果数据库没有在ODBC管理器中注册,那么它将不会在DBMS中成为一个ODBC数据源。

可以直接在Windows控制面板中的ODBC管理器中注册数据库。下面举例说明Microsoft Access数据库、Oracle数据库以及SQL Server数据库的注册方法。

### 10.1.1 为Access数据库建立ODBC数据源

当安装LINGO时,一个用于运输模型的Microsoft Access数据库就已经被默认安装了。该文件位于SAMPLES目录下,名为TRANDB.MDB。对于该运输模型,需要按如下步骤将此数据库作为ODBC数据源进行注册:

1)双击桌面上的"我的电脑"图标;

2)找到"控制面板"图标并双击它;

3)找到"ODBC数据源"图标并双击它,这时就可以看到图10-1所示的ODBC管理器窗口;

4)在ODBC管理器窗口中单击Add按钮,打开图10-2所示对话框窗口;

5)选择Microsoft Access Driver选项并且按Finish按钮,这时会出现图10-3所示对话框窗口;

6)在Data Source Name(数据源名)文本框中输入"Transportation"作为数据源的名称,在Description(描述)文本框中输入"Data source for a LINGO transportation model"为以后提醒用户该数据源的用途(也可省略不填),点击Select按钮并选择名称为"LINGO h SAMPLES h TRANDB.mdb"的数据库,这时,该对话框如图10-4所示;

7)点击OK按钮,可以看到在ODBC数据源的列表中已经增加了图10-5所示名为"Transportation"、Driver为"Microsoft Access Driver (*.mdb)"的数据源;


**图 10-1　ODBC 管理器窗口**

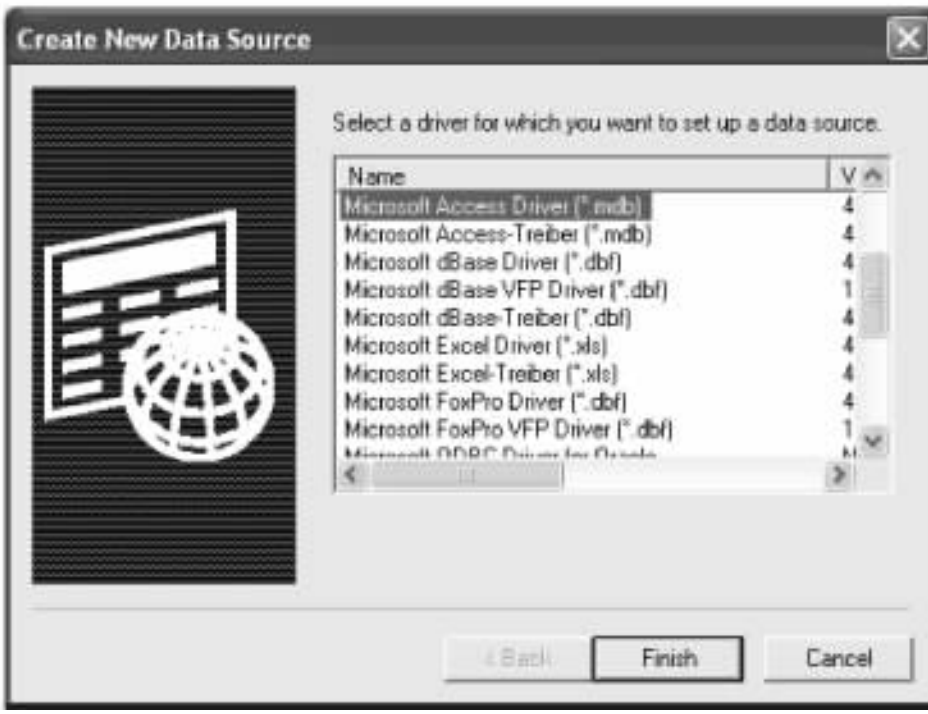



**图 10-2　创建新数据源对话框**

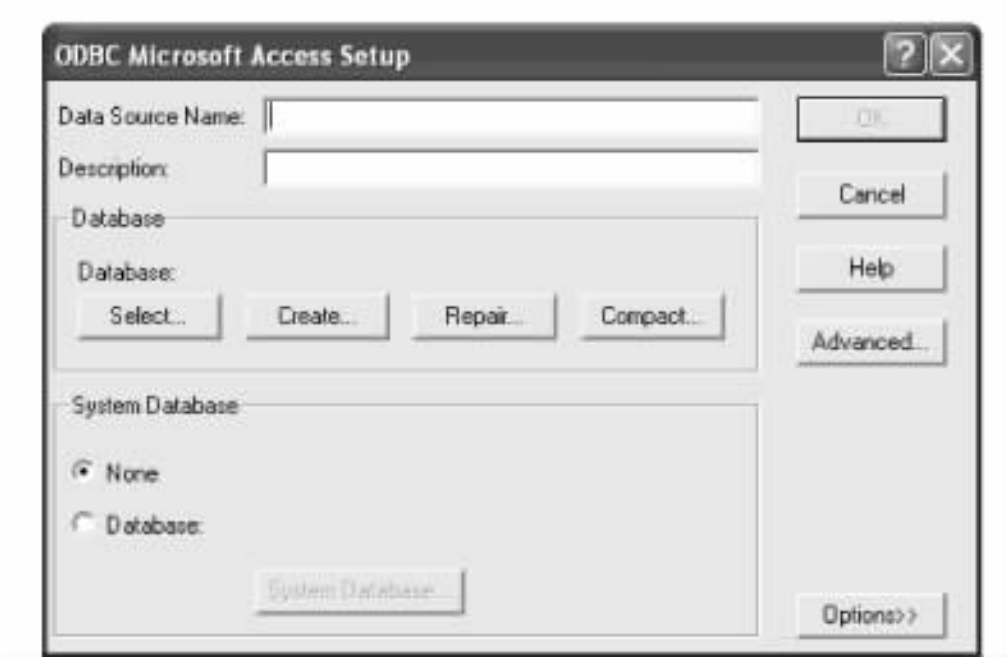



图 10-3　ODBC Microsoft Access 安装对话框

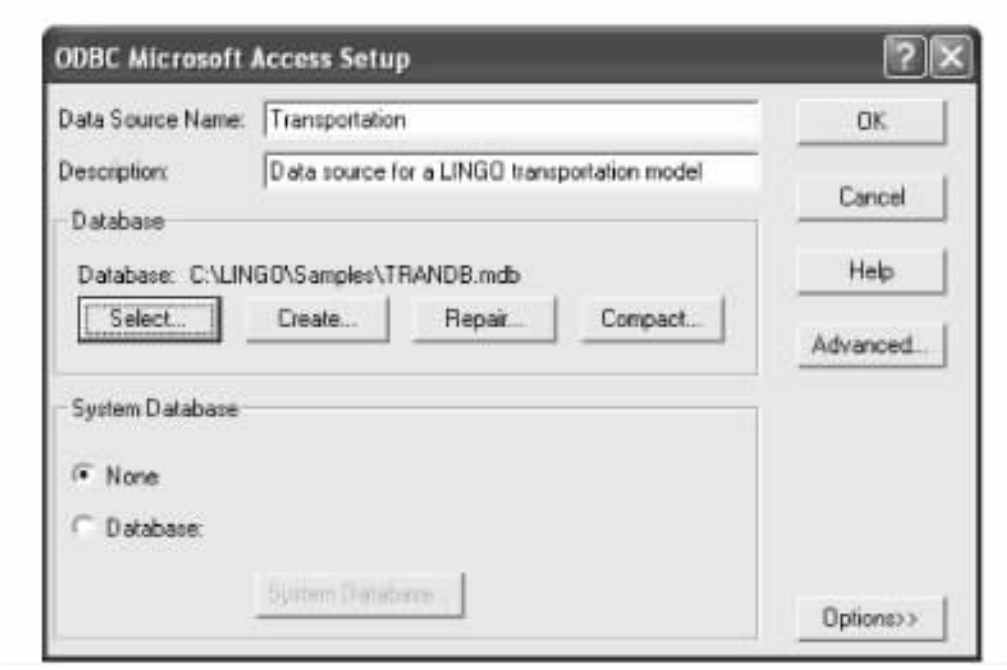



图 10-4　ODBC Microsoft Access 安装对话框

8）点击 OK 按钮关闭 ODBC 管理器。

完成以上步骤后，就可以启动 LINGO 并求解 TRANDB. LG4 模型。由于模型的名称是“Transportation”，因此 LINGO 知道模型中所需数据的数据源名称。如果运行该模型，可以看到

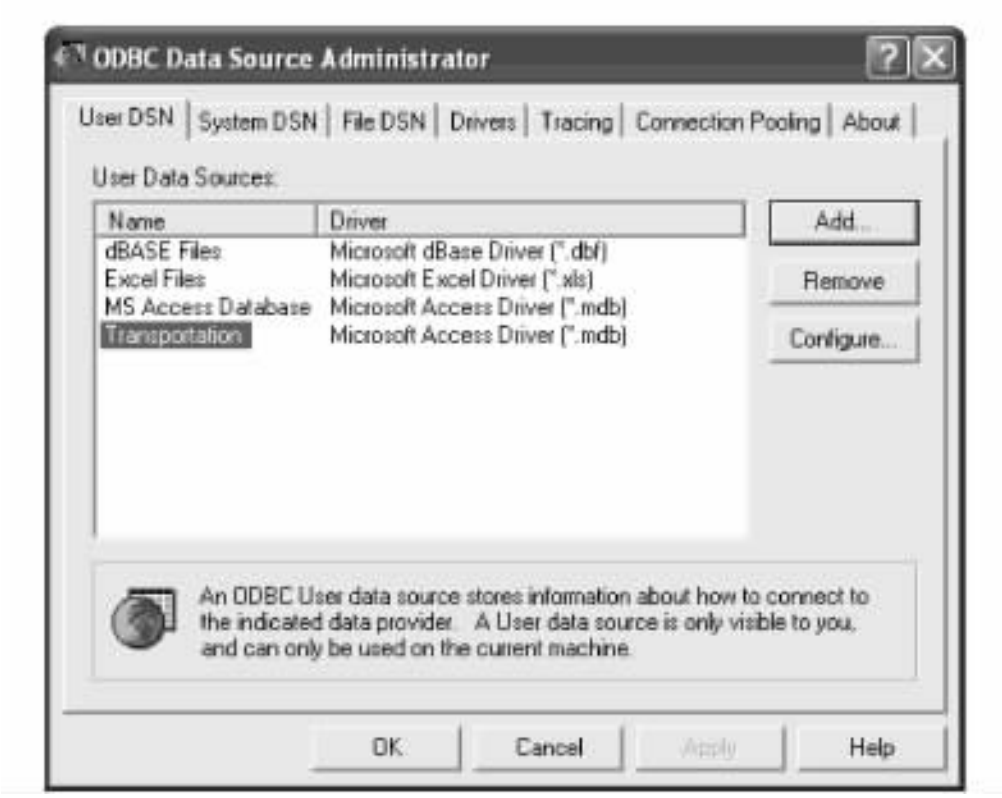



**图** 10-5　Transportation **数据源**

**如下运行结果：**

```
Global optimal solution found at step:          6
Objective value:                        161.0000
Model Title:Transportation

                      Variable           Value        Reduced Cost
                  CAPACITY(PLANT1)       30.00000     0.0000000
                  CAPACITY(PLANT2)       25.00000     0.0000000
                  CAPACITY(PLANT3)       21.00000     0.0000000
                     DEMAND(CUST1)       15.00000     0.0000000
                     DEMAND(CUST2)       17.00000     0.0000000
                     DEMAND(CUST3)       22.00000     0.0000000
                     DEMAND(CUST4)       12.00000     0.0000000
            COST(PLANT1, CUST1)          6.000000     0.0000000
            COST(PLANT1, CUST2)          2.000000     0.0000000
            COST(PLANT1, CUST3)          6.000000     0.0000000
            COST(PLANT1, CUST4)          7.000000     0.0000000
            COST(PLANT2, CUST1)          4.000000     0.0000000
            COST(PLANT2, CUST2)          9.000000     0.0000000
            COST(PLANT2, CUST3)          5.000000     0.0000000
            COST(PLANT2, CUST4)          3.000000     0.0000000
```

| | | |
|---|---|---|
| COST (PLANT3, CUST1) | 8.000000 | 0.0000000 |
| COST (PLANT3, CUST2) | 8.000000 | 0.0000000 |
| COST (PLANT3, CUST3) | 1.000000 | 0.0000000 |
| COST (PLANT3, CUST4) | 5.000000 | 0.0000000 |
| VOLUME (PLANT1, CUST1) | 2.000000 | 0.0000000 |
| VOLUME (PLANT1, CUST2) | 17.00000 | 0.0000000 |
| VOLUME (PLANT1, CUST3) | 1.000000 | 0.0000000 |
| VOLUME (PLANT1, CUST4) | 0.0000000 | 2.000000 |
| VOLUME (PLANT2, CUST1) | 13.00000 | 0.0000000 |
| VOLUME (PLANT2, CUST2) | 0.0000000 | 9.000000 |
| VOLUME (PLANT2, CUST3) | 0.0000000 | 1.000000 |
| VOLUME (PLANT2, CUST4) | 12.00000 | 0.0000000 |
| VOLUME (PLANT3, CUST1) | 0.0000000 | 7.000000 |
| VOLUME (PLANT3, CUST2) | 0.0000000 | 11.00000 |
| VOLUME (PLANT3, CUST3) | 21.00000 | 0.0000000 |
| VOLUME (PLANT3, CUST4) | 0.0000000 | 5.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| OBJ | 161.0000 | 1.000000 |
| 2 | 0.0000000 | -6.000000 |
| 3 | 0.0000000 | -2.000000 |
| 4 | 0.0000000 | -6.000000 |
| 5 | 0.0000000 | -5.000000 |
| 6 | 10.00000 | 0.0000000 |
| 7 | 0.0000000 | 2.000000 |
| 8 | 0.0000000 | 5.000000 |

在安装 LINGO 时提供了另外一个数据库，名为 TRANDB2.MDB，同样保存在 SAMPLES 目录中。可以利用该数据库重新定位 TRANDB.LG4 模型的数据来源。TRANDB.MDB 和 TRANDB2.MDB 的主要区别在于数据的维度。TRANDB.MDB 含有 3 个 PLANT 和 4 个 CUSTOMER，而 TRANDB2.MDB 含有 50 个 PLANT 和 200 个 CUSTOMER。如果在 ODBC 管理器中注册的数据源名称为“Transportation2”，而且相应的数据库名称为“TRANDB2.MDB”，同时用“TITLE Transportation2;”语句将该运输模型的名称改为“Transportation2”，这样就可以将 LINGO 重新定位而使用新的数据源了。

这个新的模型将含有 10 000 个变量，而原先的模型仅仅有 12 个变量。这种轻松运行不同数据规模集合的能力说明了编写具有独立数据、基于集合的模型的用处。

## 10.1.2 为 Oracle 数据库建立 ODBC 数据源

LINGO 主目录的 SAMPLES 文件夹中有一个名为 TRANDB.SQL 的 SQL 脚本文件，可以通过 Oracle 的 SQL Plus 功能运行该文件，从而为运输模型 TRANDB.LG4 建立一个小型数据库。以下是建立这个数据源需要完成的步骤。

1）通过 Windows 的开始菜单，启动 Oracle 提供的 SQL Plus 功能，选择“运行”命令，输入“SQLPLUSW”并且点击“确定”按钮。

2）输入 Oracle 的用户名和密码。如果使用的是默认 Oracle 安装，那么用户名是“sys”，密码是“change _ on _ install”。只有当 Oracle 远程运行时才需要主机名。

3）在 SQL Plus 系统提示符后输入“@C：h LINGO8 h SAMPLES h TRANDB. SQL”运行 TRANDB. SQL 脚本。

4）系统提示符后输入“EXIT”以退出 SQL Plus。

5）按照前面所描述的方法启动 ODBC 管理器。

6）当选择 ODBC 数据源驱动程序时，确保使用的是“Microsoft ODBC for Oracle”，而不能使用“Oracle ODBC Driver”，因为后者不能提供所有需要的功能。

7）指定数据源的名称为“Transportation”。

8）在 LINGO 中打开并运行模型文件 TRANDB. LG4。

9）按提示输入 Oracle 用户名和密码。如果希望 LINGO 访问数据库时不必每次都输入 Oracle 用户名和密码，可以在 LINGO 运行之初配置 File 菜单中的 Database User Info 选项。

### 10.1.3 为 SQL Server 数据库建立 ODBC 数据源

介绍这部分内容之前，首先需要介绍如何将 LINGO 默认安装的 Access 数据库文件 TRANDB. MDB 中的数据导入到 Microsoft SQL Server 中。将数据从 Microsoft Access 复制到 Microsoft SQL Server 的步骤如下。

1）通过桌面的“开始”菜单，在“程序”选项中选择 Microsoft SQL Server 的“导入和导出数据”功能。

2）打开“DTS 导入/导出向导”对话框，点击“下一步”，则出现图 10-6 所示对话框。在“数据源”选项中选择“Microsoft Access”，在“文件名”选项中选择 TRANDB. mdb（根据实际安装目录而定）。

3）点击“下一步”则出现图 10-7 所示对话框。

4）在该对话框中需要确定数据复制的目的地。“目的”选项选择“用于 SQL Server 的 Microsoft ODBC 驱动程序”；“服务器”选项选择服务器名称；使用“服务器”下面的单选择按钮选择身份验证方式。假定已经在 SQL Server 的“企业管理器”中创建了一个名称为“Lingo 运输”的数据库，在“数据库”下拉列表中选择“Lingo 运输”，这样 TRANDB. mdb 数据库中的所有 Access 数据表将被复制到 SQL Server 的“Lingo 运输”数据库中。

5）点击“下一步”出现图 10-8 所示对话框，选择“从源数据库复制表和视图”。

6）点击“下一步”出现图 10-9 所示对话框。由于要把 TRANDB. mdb 数据库中的所有 Access 数据表复制到 SQL Server 的“Lingo 运输”数据库中，所以将三个源表“Arcs”、“Customers”、“Plants”全部选中，点击“下一步”，通过一系列确定信息后，就成功地将三个数据表从 Microsoft Access 复制到 Microsoft SQL Server 中。当再次打开 Microsoft SQL Server 的企业管理器时，可以看到包含有“Arcs”、“Customers”、“Plants”三个用户数据表的“Lingo 运输”数据库。

需要注意的是，从 Microsoft Access 复制到 Microsoft SQL Server 中的数据表的字段类型有时需要作一定调整。例如，如果某字段是 nvarchar 类型，一般需要将其改为 char 类型，长度根据用户需要而定，这样才能使 LINGO 模型正常运行。

通过以上步骤，已经将数据从 Microsoft Access 复制到了 Microsoft SQL Server 中。这样就按


**图 10-6　DTS 导入/导出向导对话框|选择数据源**


**图 10-7　DTS 导入/导出向导对话框|选择目的**

**照上文介绍的步骤使Microsoft SQL Server中的“Lingo 运输’数据库成为了一个 ODBC 数据源。**


**图 10-8　DTS 导入/导出向导对话框|指定表复制或查询**


**图 10-9　DTS 导入/导出向导对话框|选择源表和视图**

## 10.2 利用@ODBC函数从数据库中导入数据

### 10.2.1 利用@ODBC函数导入数据语法

为了能够从ODBC数据源中导入模型所需要的数据,需要在模型的数据域使用@ODBC函数。@ODBC函数允许导入文本格式的集合元素以及用数字表示的集合属性值。@ODBC函数导入数据的语法如下:

```
object_list = @ODBC([data_source' [,' table_name'
  [,' column_name_1' [,' column_name_2' ...]]]]);
```

其中,object_list是一个对象列表,各对象之间可以由逗号分隔,包含由ODBC数据源初始化的模型对象(属性、集合元素、变量等)。object_list至多可以包含一个集合(基本集合或衍生集合)的名字以及多个集合属性,在object_list中的所有集合属性都必须在同一个集合中定义,并且,如果object_list中包含一个集合的名字,那么在object_list中的所有属性都必须是在该集合中定义的。data_source参数是用ODBC管理器注册过的ODBC数据源的名称。table_name参数是在该数据源中数据表的名称。column_name参数是数据表中的列名,即字段名。初始化集合属性或基本集合需要一列数据,而初始化衍生集合需要一列数据对应集合的一维。因此,一个二维的衍生集合需要两个数据列来初始化它的集合元素。

如果省略了data_source(即数据源参数),那么在该模型文件中应该使用TITLE命令指明模型的标题。

如果省略了table_name(即数据表名),那么在object_list中的集合的名称将取而代之。如果在object_list中没有集合的名称,那么在object_list中所列出的集合属性所对应的集合名称将代替省略的数据表名。

如果省略了column_name(即列名参数),那么LINGO将基于object_list中相应的对象是一个集合属性或一个基本集合还是一个衍生集合来选择默认列名。若要初始化的对象是一个集合属性或者是一个基本集合,LINGO将使用集合名作为默认的列名。若要初始化的对象是一个衍生集合,LINGO将为衍生集合的每一维生成一个默认的列名,并且列名与衍生对应维的基本集合名称相同。例如,一个名为LINKS的二维集合衍生于两个基本集合SOURCE和DESTINATION,它将从默认名为SOURCE和DESTINATION的两列中得到初始数据。

只有在省略列名参数的情况下才可以省略数据表名参数,也只有在数据表名参数和列名参数都被省略的情况下才可以省略数据源参数。

需要注意的是,在数据源里用于产生基本集合元素的数据必须是文本格式,数字格式的数据不能用于生成集合元素。在数据源里用于产生集合属性的数据也必须是数字格式。

在模型的数据域中用@ODBC导入数据的一些例子如下。

例1:

```
SHIPPING_COST =
  @ODBC('TRANSPORTATION','LINKS','COST');
```

此例中，LINKS 数据表存放在 ODBC 数据源 TRANSPORTATION 中，LINGO 将此表的 COST 列中的元素导入到基本集合 SHIPPING _ COST 中（假设 SHIPPING _ COST 是基本集合的名称）。

例 2：

```
VOLUME, WEIGHT =
@ODBC('TRUCKS', 'CAPACITY');
```

这个例子中，数据库的列名被省略了。因此，若假设 VOLUME 和 WEIGHT 是集合属性，那么 LINGO 将默认使用属性名 VOLUME 和 WEIGHT 为列名。具体地讲，CAPACITY 数据表存放在 ODBC 数据源 TRUCKS 中，LINGO 用此表的 VOLUME 和 WEIGHT 列中的元素数据初始化 VOLUME 和 WEIGHT 这两个属性。

例 3：

```
REQUIRED, DAYS = @ODBC();
```

这个例子中，假设已将模型命名为 PRODUCTION，REQUIRED 是从两个基本集合 JOB 和 WORKSTATION 衍生出来的一个衍生集合（例如，REQUIRED（JOB，WORKSTATION）），DAYS 是一个定义在 REQUIRED 集合上的属性。

在 @ODBC 函数中省略了所有的参数，所以 LINGO 将 PRODUCTION 作为数据源名，将集合名 REQUIRED 作为数据表名，将 JOB、WORKSTATION 和 DAYS 作为三个列名。如果想把该语句书写完整，可以用以下同样功能的语句将所有的参数包含在 @ODBC 函数中：

```
REQUIRED, DAYS = @ODBC('PRODUCTION', 'JOB', 'WORKSTATION', 'DAYS');
```

## 10.2.2 在 PERT 模型中利用 @ODBC 函数导入数据

下面将修改第 2 章中的 PERT 模型，以说明如何利用 @ODBC 函数从一个 Microsoft Access 数据库中导入工程项目名称（作为集合元素），修改部分用黑体字表示。模型如下：

```
SETS:
  TASKS:TIME, ES, LS, SLACK;
  PRED(TASKS, TASKS);
ENDSETS
DATA:
  TASKS = @ODBC('PERTODBC', 'TASKS', 'TASKS');
  PRED = @ODBC('PERTODBC', 'PRECEDENCE',
    'BEFORE', 'AFTER');
  TIME = @ODBC('PERTODBC');
ENDDATA
@FOR(TASKS(J)|J #GT# 1:
   ES(J) = @MAX(PRED(I, J):ES(I) + TIME(I))
);
@FOR(TASKS(I)|I #LT# LTASK:
LS(I) = @MIN(PRED(I, J):LS(J) - TIME(I));
```

```
);
    @FOR(TASKS(I):SLACK(I) = LS(I) - ES(I));
    ES(1) = 0;
    LTASK = @SIZE(TASKS);
    LS(LTASK) = ES(LTASK);
```

首先,通过语句"TASKS = @ODBC('PERTODBC','TASKS','TASKS');"就可以从 ODBC 数据源 PERTODBC 中获得 TASKS 集合的集合元素。数据源 PERTODBC 中有一个 TASKS 数据表,数据表中有一个列名为 TASKS 的列,正是从这个列中获取集合 TASKS 的元素。图 10-10 是在 Access 中的 TASKS 数据表。


图 10-10 PERTODBC.MDB|TASKS 数据表

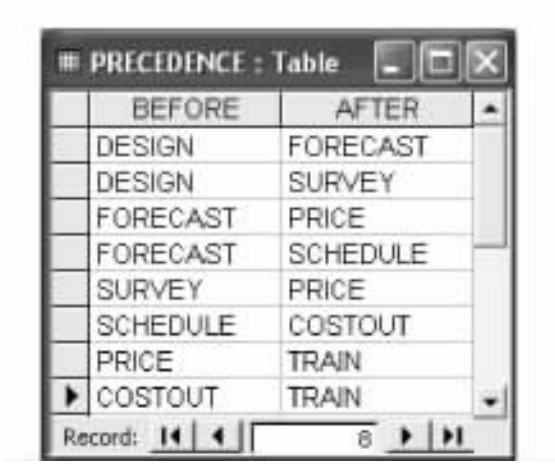



图 10-11 PERTODBC.MDB|PRECEDENCE 数据表

然后,用语句"PRED = @ODBC('PERTODBC','PRECEDENCE','BEFORE','AFTER');"从 ODBC 数据源中得到 PRED 集合的元素。具体地讲,PRECEDENCE 数据表位于 PERTODBC 数据源中,用 PRECEDENCE 数据表中的 BEFORE 和 AFTER 两列形成衍生集合 PRED 的集合元素。图 10-11 是在 Access 中表示该模型中工程任务优先关系的数据表。

为了得到工程任务时间的值 TIME,创建了一个 ODBC 连接:

```
TIME = @ODBC('PERTODBC');
```

可见,在该@ODBC 函数中只指定了 ODBC 数据源的名称,而表名和列名都被省略了。在这种情况下,LINGO 为数据表和列提供了默认值。TIME 是一个集合属性,所以 LINGO 将它所属的基本集合名 TASKS 作为默认的数据表名称,并且将集合属性名 TIME 作为默认的列名。也就是说,数据表 TASKS 中的名为 TIME 列中的数据用于初始化属性 TIME,数据表 TASKS 存放在数据源 PERTODBC 中。如果希望把该语句书写完整,可以用以下方式将所有的参数包含在@ODBC 函数中:

```
TIME = @ODBC('PERTODBC','TASKS','TIME');
```

上面 PERT 模型里的所有数据(集合元素和属性值)都存放在 ODBC 数据源里,所有的数据都是通过@ODBC 函数输入的。

## 10.3 利用@ODBC函数导出数据

### 10.3.1 利用@ODBC函数导出数据的语法

与很多接口函数一样，@ODBC函数既可以导入数据，也可以导出数据。在模型数据域中，可以利用@ODBC函数将集合元素和属性值导出到ODBC数据源中。利用@ODBC函数导出数据的语法如下：

```
@ODBC([data_source'[,'table_name'[,'column_name_1'[,
  'column_name_2'...]]]]) = object_list;
```

其中各参数的含义及注意事项同10.2节所述。

在模型的数据域使用@ODBC函数将数据导出到ODBC数据源的一些例子如下。

例1：

```
@ODBC('TRANSPORTATION',
  'LINKS','VOLUME') = VOLUME;
```

在这个例子中，LINGO将VOLUME属性值导出到LINKS数据表的VOLUME列，而LINKS数据表是在TRANSPORTATION数据源中。

例2：

```
@ODBC() = NUMBER_WORKING;
```

这个例子中，@ODBC函数中所有的参数都被省略了，LINGO默认将模型的标题作为数据源名，将定义属性的集合名作为数据表名，并且将属性名作为列名。假设已经在模型SCHEDULING中使用了TITLE命令将模型的标题定义为SCHEDULING，并且属性NUMBER_WORKING已经定义在集合SCHEDULES中，那么LINGO将NUMBER_WORKING属性值导出到SCHEDULES数据表的名称同样为NUMBER_WORKING的列中，而数据表SCHEDULES存在于SCHEDULING数据源中。

### 10.3.2 在PERT模型中利用@ODBC函数导出数据

继续使用上节中的PERT模型，通过一定的修改将最早开始时间和最迟开始时间（ES和LS）的求解结果值导出到PERTODBC数据源中。

首先，建立一个名为SOLUTION的数据表，该表中有三个数据列，分别为TASKS、EARLIEST START和LATEST START。接收TASKS集合元素值的数据列（即TASKS字段）应该是文本格式，而接收ES和LS属性值的列EARLIEST START和LATEST START应该是数字格式。图10-12是存放导出数据的SOLUTION表。

通过修改模型，可以将数据导出到SOLUTION表中（修改部分用黑体字体表示），模型如下：

```
SETS:
```

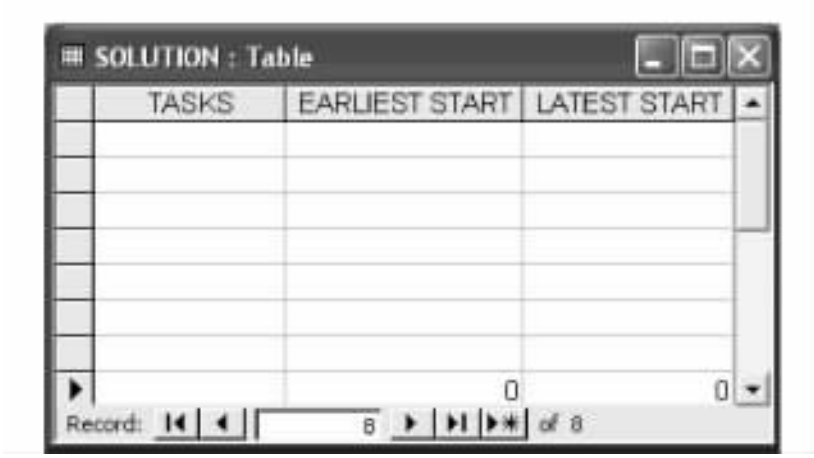



图 10-12　PERTODBC.MDB|SOLUTION 数据表

```
    TASKS:TIME, ES, LS, SLACK;
    PRED(TASKS, TASKS);
  ENDSETS
  DATA:
    TASKS = @ODBC('PERTODBC', 'TASKS', 'TASKS');
    PRED = @ODBC('PERTODBC', 'PRECEDENCE', 'BEFORE', 'AFTER');
    TIME = @ODBC('PERTODBC');
    @ODBC('PERTODBC', 'SOLUTION', 'TASKS',
    'EARLIEST START', 'LATEST START') =
    TASKS, ES, LS;
  ENDDATA
  @FOR(TASKS(J)|J #GT# 1:
    ES(J) = @MAX(PRED(I, J):ES(I) + TIME(I))
);
  @FOR(TASKS(I)|I #LT# LTASK:
  LS(I) = @MIN(PRED(I, J):LS(J) - TIME(I));
);
  @FOR(TASKS(I):SLACK(I) = LS(I) - ES(I));
  ES(1) = 0;
  LTASK = @SIZE(TASKS);
  LS(LTASK) = ES(LTASK);
```

该模型运行求解后,更新的 SOLUTION 数据表如图 10-13 所示。

## 10.3.3　输出统计报告

用@ODBC 函数实现 LINGO 与数据库之间的数据交换,就会在求解结果报告窗口的顶部生成输出统计报告。对应于模型中所使用的每一个@ODBC 语句,都会出现一个这样的报告。该报告列出关于@ODBC 数据导出操作的详细资料。在 PERT 模型中,可以看到如下统计报

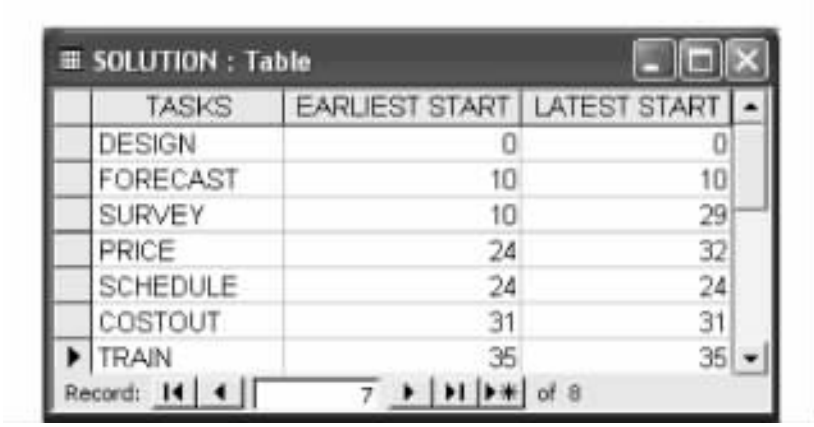



**图** 10-13　PERTODBC. MDB | SOLUTION **数据表**

**告：**

```
Export Summary Report
- - - - - - - - - - - - - - - - - - - - -
Transfer Method:       ODBC BASED
ODBC Data Source:      PERTODBC
Data Table Name:       TASKS
Columns Specified:      3
  TASKS
  EARLIEST
  LATEST
LINGO Column Length:              7
Database Column Length:           7
```

**当在模型中利用** @ODBC **函数导出数据时，报告中的** Transfer Method **(传输方法) 域总是“**ODBC BASED**”。其次，数据源名和数据表名分别在** ODBC Data Source **(**ODBC **数据源) 域和** Data Table Name **(数据表名称) 域中列出，**Columns Specified **域中列出该数据表中列的个数以及各列的名称。**LINGO Column Length **域中显示的是每个属性元素中的个数。**Database Column Length **域中显示数据库中接收列的长度。总之，**LINGO Column Length **必须和** Database Column Length **一致，否则，**LINGO **要么会截断它的输出，要么在数据库中会没有足够的数据填满接收列。如果** LINGO **处于紧凑输出模式，那么不会出现输出统计报告。**

# 第 11 章　与其他应用软件的接口

LINGO 有着便捷、交互式的界面以及庞大的、可即时调用的函数库，这对建立和解决数学模型大有裨益。然而，用户有时可能希望将 LINGO 强大的功能集成到自己开发的应用软件内部，或者在其他外部编程语言中调用 LINGO 模型中的函数。在 Windows 系统中，LINGO 通过调用 Windows 下的动态链接库（以下简称 DLL）实现了与用户自己定制的应用程序的关联。通过建立与 LINGO 挂接的应用程序，可以让使用者在不熟悉任何 LINGO 模型细节的前提下，在简单的模式中输入数据并查看结果，以体验 LINGO 的强大功能。而且，应用程序在后台调用 LINGO 进行计算，不会被使用者察觉。另外，LINGO 还能将一些常用的函数定制成 DLL 文件，并通过 @USER 命令在任何模型中调用前面定制的函数。

## 11.1　LINGO 动态链接库

Windows 版本的 LINGO 有一个可即时调用的动态链接库（Dynamic Linked Libraries，DLL）。访问 DLL 是所有 Windows 开发环境的基本功能，如 Visual Basic、Delphi 和 Visual C++ 等。LINGO 提供的 DLL 是 32 位的，可以在目前所有 Windows 常用的版本下运行。

本节介绍在 Visual C/C++、Visual Basic 以及 Delphi 中调用 LINGO DLL 的详细方法。采用其他开发语言的用户也会对本章的例子感兴趣，因为其中的许多思路和方法在其他的开发环境下同样能够适用。

与 LINGO DLL 的接口技术相对比较简单，用于实现在用户的应用程序中运行 LINGO 脚本命令，这样就能够通过脚本命令获得 LINGO 所有的重要特性。然而，用户必须非常熟悉 LINGO 的命令语言，这样才能建立起有用的脚本文件。

注意　如果在 LINGO 的交互版本中没有输入系统密码，那么在运行软件第一次调用 LINGO DLL 时，系统将会提示输入密码。此后若使用同一台机器调用 LINGO DLL 不再需要输入密码。

### 11.1.1　在员工安排模型中应用 LINGO DLL

为了说明如何使用 LINGO DLL 接口，再次使用 Pluto Dogs 公司员工安排的例子。在这部分内容中，用 Visual C++ 和 Visual Basic 编程，设计图 11-1 所示界面。

在上述界面中，用户在 Needs 列的文本框中输入员工的需求量，点击 Solve 按钮，应用程序首先从这个界面中获取员工需求量，然后把这些值随模型一起传给 LINGO DLL 求解，最后将结果传回该界面，以便得到可视化的结果。具体地说，Start 列中显示每天开始工作的员工数，On Duty 列中显示当班的员工数，Total 文本框中显示员工需求的总数。

运行该模型后，窗口显示的结果如图 11-2 所示。

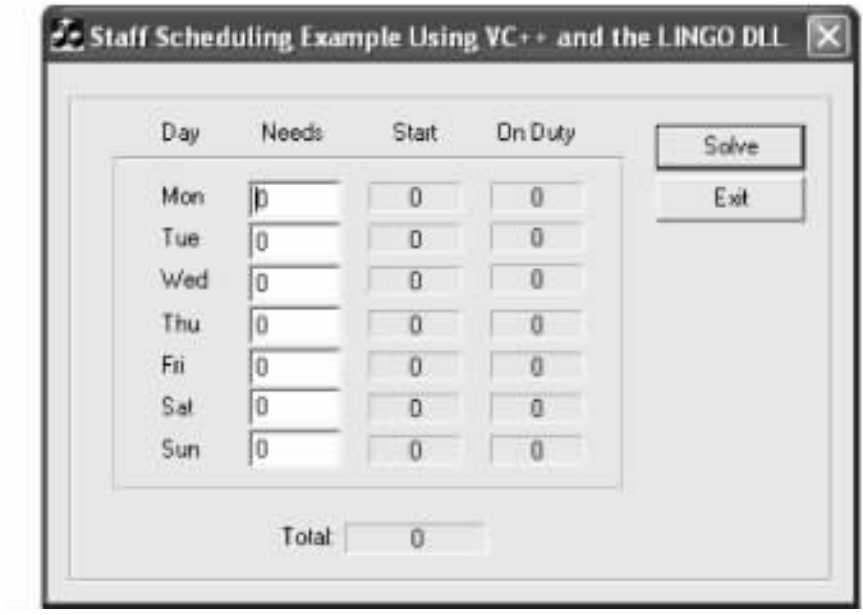



**图 11-1　员工安排模型界面**

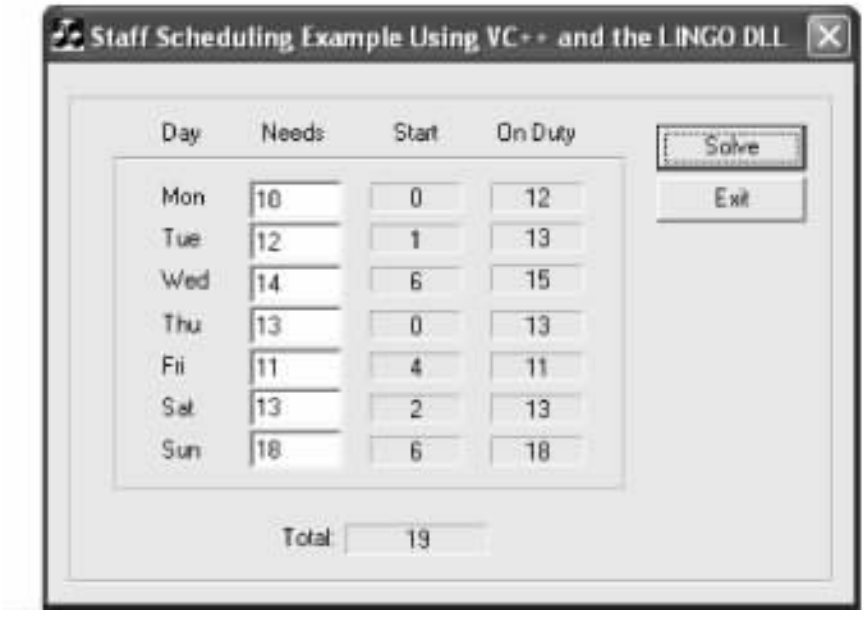



**图 11-2　员工安排模型求解结果**

## 11.1.2　模型

**在上面的例子中，通过 LINGO DLL 传送到 LINGO 中求解的员工安排模型类似于以前用过的模型，但是作了一些修改，修改的部分以黑体字表示。模型如下：**

```
MODEL:
```

```
  SETS:
    DAYS / MON TUE WED THU FRI SAT SUN/:
      NEEDS, START, ONDUTY;
  ENDSETS
  [OBJECTIVE] MIN = @SUM(DAYS(I):START(I));
  @FOR(DAYS(TODAY):
  !Calculate number on duty;
    ONDUTY(TODAY) =
      @SUM(DAYS(D)|D #LE# 5:
        START(@WRAP(TODAY - D + 1,
          @SIZE(DAYS))));
  !Enforce staffing requirement;
    ONDUTY(TODAY) >= NEEDS(TODAY);
    @GIN(START);
  );
  DATA:
    NEEDS = @POINTER(1);
    @POINTER(2) = START;
    @POINTER(3) = ONDUTY;
    @POINTER(4) = OBJECTIVE;
    @POINTER(5) = @STATUS();
  ENDDATA
END
```

由于 LINGO 的脚本处理器要读取这个模型，因此模型必须以“MODEL:”命令开始，以“END”命令结束。为 DAYS 集合添加 ONDUTY 属性，用来计算每天当班的员工数量，以便将它们传回到应用程序并放到界面上的 On Duty 列中。将目标函数命名为 OBJECTIVE，目的同样是将目标值返回应用程序并显示在 Total 文本框中。

### 11.1.3 @POINTER 函数

在数据域中，@POINTER 函数作为应用程序和 LINGO 求解器的内存链接，功能非常强大，负责 LINGO 与应用程序之间的数据交换。通过 @POINTER 函数可以进行数据的快速传输，而无需基于磁盘文件存储空间的建立和分解。

当调用 LINGO DLL 时，可以传送给它一个内存指针的列表。@POINTER(n) 指向所传送的指针列表中的第 n 个指针。只有在模型中使用 DLL 或 OLE 接口访问 LINGO 时，@POINTER 函数才有意义。

如果 @POINTER 函数出现在数据陈述语句的右边，如下面的情况：

```
NEEDS = @POINTER(1);
```

LINGO 将对表达式左边的集合属性进行初始化，数据则来自于陈述语句右边的指针所指位置开始的内存区域。相反，当 @POINTER 函数出现在陈述语句的左边时，LINGO 将通过数据陈述

语句左边的指针把在陈述语句右边的属性值传送到内存区域中。

利用@POINTER 函数读入和写出的所有数据均为双精度浮点型。BASIC 和 C/C++认为这是双精度型(DOUBLE),而 FORTRAN 把它归为 8 位实型(REAL*8)或双精度型(DOUBLE PRECISION)。

当设置@POINTER 函数指向的存储区域时,必须确保已分配足够的空间。LINGO 假设所给的存储区域有足够的空间并且提供的数据在此区域均可得到。如果没有分配足够的空间或可用值,可能会出现存储保护错误,更严重的是,可能会出现没有警告的错误结果。

重新看一下该模型的数据域:

```
DATA:
  NEEDS = @POINTER(1);
  @POINTER(2) = START;
  @POINTER(3) = ONDUTY;
  @POINTER(4) = OBJECTIVE;
  @POINTER(5) = @STATUS();
ENDDATA
```

可以看到,员工需求属性 NEEDS 是从第一个存储区域读取的,START、ONDUTY 和 OBJECTIVE 的值则分别被写入第二、第三和第四个存储区域。最后,@STATUS 函数的值被写入第五个存储区域。

## 11.1.4 LINGO DLL 的输出函数

### 11.1.4.1 输出函数

下面列出 LINGO DLL 的十个输出函数。用户可以用库文件 h LINGO8 h DLL h LINGD80.LIB 将这些函数引入 Microsoft C 或者 FORTRAN 程序中。Visual Basic 用户可以通过 h LINGO8 h DLL h LINGD80.BAS 模块文件来引入这些函数。Borland Delphi 用户可以通过 h LINGO8 h DLL h Lingodel.pas 模块文件来引入这些函数。其他开发环境下的用户需要通过 LINGO 的 DEF 文件 h LINGO8 h DLL h LINGD80.DEF 构建自己的引入库,通过 DEF 文件创建引入库的方法可以参考该程序环境的帮助文档。

被输出的函数包含在 LINGD80.DLL 的文件中,该文件作为 LINGO 安装的一部分被安装在 Windows 的系统文件夹中。LINGD80.DLL 需要许多附加 DLL 文件,这些文件在正常安装 LINGO 时均被安装。如果需要特别列出 LINGD80.DLL 所需要的支撑 DLL 文件,可以用 Dependency Walker 打开 LINGD80.DLL 文件查看。免费下载 Dependency Walker 的网址是:http://www.dependencywalker.com。Dependency Walker 程序是一个非常有效的工具,用于列出应用软件所需要的 DLL 文件以及这些文件在系统中的位置,并且标记出目前无法获得的 DLL 文件。

下面简要说明 LINGO DLL 的输出函数及每个函数的功能。函数定义是按 C 语言习惯编写的。Visual Basic 程序员可以参阅 h LINGO8 h DLL h LINGD80.BAS,以查看使用 Basic 语言编写的函数定义;Borland Delphi 程序员可以参阅 h LINGO8 h DLL h Lingodel.pas,以查看使用 Delphi 语言编写的函数定义。

1. void LSclearPointersLng (pLSenvLINGO pL)

该函数用于清除@POINTER()指针列表。这些指针指向通过调用 LSsetPointerLng()函数而建立的存储传输区域。

参数意义：

pL——指向 LINGO 环境的指针，该指针由先前调用的 LScreateEnvLng()函数创建。

2. int LScloseLogFileLng (pLSenvLINGO pL)

该函数用于关闭 LINGO 的日志文件。该日志文件由先前调用的 LSopenLogFileLng()函数打开。

参数意义同前。

返回值：

如果没有错误返回 0，否则将返回一个非零错误代码。

3. pLSenvLINGO CALLTYPE LScreateEnvLng()

该函数用于创建 LINGO 环境对象，所有其他 LINGO DLL 函数都需要一个有效的指针指向一个 LINGO 环境对象，并且应该在程序末尾通过调用 LSdeleteEnvLng()函数释放这个对象。

返回值：

如果发生错误返回 0，否则返回一个指向 LINGO 环境对象的指针。

4. int LSdeleteEnvLng (pLSenvLINGO pL)

该函数用于删除先前通过调用 LScreateEnvLng()函数创建的 LINGO 环境对象，并释放分配给 LINGO 对象的系统内存。

参数意义同前。

返回值：

如果没有错误返回 0，否则将返回一个非零错误代码。

5. int LSexecuteScriptLng (pLSenvLINGO pL, char * pcScript)

该函数主要用来处理 LINGO 命令脚本，是 LINGO DLL 的主要功能函数。脚本可能全部存放在内存中，或者通过一个或更多的 TAKE 命令从磁盘中加载脚本。

参数意义：

pcScript——指向包含 LINGO 命令脚本的字符串的指针，脚本的每行必须以换行符 (ASCII 10) 结束，整个脚本必须以 NULL (ASCII 0) 终止。

其他参数意义同前。

返回值：

如果没有错误返回 0，否则将返回一个非零错误代码。

6. int LSgetCallbackInfoLng (pLSenvLINGO pL, int nObject, void * pResult)

用户可以在自己的应用程序中创建一个函数，LINGO 求解器定期访问该函数以使用户随时了解 LINGO 的求解进度，这类函数被称为回调函数。回调函数可以通过 LSgetCallbackInfoLng()函数访问 LINGO，从正在处理模型的求解器中得到特定的信息。

参数意义：

nObject——查找信息的索引，其可能值如表 11-1 所示；

表 11-1　信息索引

| 索引 | 名称 | 类型 | 信息 |
|---|---|---|---|
| 0 | LS _ IINFO _ VARIABLES _ LNG | Int | 变量总个数 |
| 1 | LS _ IINFO _ VARIABLES _ INTEGER _ LNG | Int | 整型变量个数 |
| 2 | LS _ IINFO _ VARIABLES _ NONLINEAR _ LNG | Int | 非线性变量个数 |
| 3 | LS _ IINFO _ CONSTRAINTS _ LNG | Int | 约束总个数 |
| 4 | LS _ IINFO _ CONSTRAINTS _ NONLINEAR _ LNG | Int | 非线性约束个数 |
| 5 | LS _ IINFO _ NONZEROS _ LNG | Int | 全部非零矩阵元素个数 |
| 6 | LS _ IINFO _ NONZEROS _ NONLINEAR _ LNG | Int | 非线性非零矩阵元素个数 |
| 7 | LS _ IINFO _ ITERATIONS _ LNG | Int | 迭代次数 |
| 8 | LS _ IINFO _ BRANCHES _ LNG | Int | 分支数目 (仅应用于整数规划问题) |
| 9 | LS _ DINFO _ SUMINF _ LNG | Double | 不可行解的总和 |
| 10 | LS _ DINFO _ OBJECTIVE _ LNG | Double | 目标值 |
| 11 | LS _ DINFO _ MIP _ BOUND _ LNG | Double | 目标边界 (仅应用于整数规划问题) |
| 12 | LS _ DINFO _ MIP _ BEST _ OBJECTIVE _ LNG | Double | 目前目标函数最优值 (仅应用于整数规划问题) |

pResult——指向 LINGO 存储用户所提问题答案的位置的指针，LINGO 将在这个地址保存问题的答案，并且必须预先为整型变量分配 4 个字节的内存，为双精度型变量分配 8 个字节的内存。

其他参数意义同前。

返回值：

如果没有错误返回 0，否则将返回一个非零错误代码。

7. int LSopenLogFileLng ( pLSenvLINGO pL, char * pcLogFile)

该函数用于为 LINGO 创建一个写入运行脚本的日志文件。一般情况下，应该在程序开始运行时就创建一个日志文件以协助调试。如果发生一个错误而不十分确定原因，这时最好的办法就是参阅日志文件寻找线索。

参数意义：

pcLogFile——指向包含日志文件路径名的字符串的指针。

其他参数意义同前。

返回值：

如果没有错误返回 0，否则将返回一个非零错误代码。

8. int LSsetCallbackErrorLng ( pLSenvLINGO pL, lngCBFuncError _ t pcbf, void * pUserData)

该函数用于指定一个回调函数，当发生错误时，LINGO 可以随时调用此函数。这使得应用程序在处理命令脚本的过程中，可以密切观察发生的非正常情况。

参数意义：

pcbf——指向回调函数的指针；

pUserData——用户设定的指针，指向回调函数中任何需要参阅的数据，LINGO 通过回调函

数只传送指针的值。所以,如果不再需要指针,可以设置其值为 NULL。

返回值:

如果没有错误返回 0,否则将返回一个非零错误代码。

用户提供的回调函数必须采用标准的调用习惯,并且具有如下的接口:

int MyErrCallback(pLSenvLINGO pL, void * pUserData, int nErrorCode, char * pcErrorText);

如果没有严格遵守接口规则,计算机极有可能瘫痪。LINGO 的错误代码将通过 nErrorCode 参数报告,同时在 pcErrorText 中存储错误文本,并且至少为错误文本设置 200 个字节的空间。

9. int LSsetCallbackSolverLng(pLSenvLINGO pL, lngCBFuncError_t pcgf, void * pUserData)

该函数用于设定一个 LINGO 求解模型时定时调用的回调函数。

参数意义同前。

返回值:

如果没有错误返回 0,否则将返回一个非零错误代码。

用户提供的回调函数必须采用标准的调用习惯,并且具有如下的接口:

int MySolverCallback(pLSenvLINGO pL, int nReserved, void * pUserData);

如果没有严格遵守接口规则,计算机极有可能瘫痪。nReserved 参数被保留以备后用,也可以被省略。

10. int CALLTYPE LSsetPointerLng(pLSenvLINGO pL, double * pdPointer, int * pnPointersNow)

一次或多次调用该函数用于将内存指针列表传递给 LINGO。LINGO 的@POINTER()函数使用这些指针,用来输入数据到求解器和从求解器输出结果。换句话说,这个程序使 LINGO 与内存之间有一个快速方便的链接进行数据传输。

参数意义:

pdPointer ——@POINTER()函数的一个实例使用的指向内存传输位置的指针;

pnPointersNow ——指向一个整型变量的指针,该整型变量为 LINGO 返回的列于@POINTER()指针列表中通道的当前数量。第一次调用 LSsetPointersLng()时,这个指针将返回 1。通过访问 LSclearPointersLng(),指针列表可以随时被清除。

返回值:

如果没有错误返回 0,否则将返回一个非零错误代码。

#### 11.1.4.2 错误代码

多数 LINGO DLL 函数都会返回一个错误代码。表 11-2 列举了所有可能的代码。

表 11-2 错误代码

| 值 | 名 称 | 描 述 |
|---|---|---|
| 0 | LSERR_NO_ERROR_LNG | 没有错误 |
| 1 | LSERR_OUT_OF_MEMORY_LNG | 超出动态系统内存 |
| 2 | LSERR_UNABLE_TO_OPEN_LOG_FILE_LNG | 无法打开日志文件 |

续表

| 值 | 名 称 | 描 述 |
|---|---|---|
| 3 | LSERR _ INVALID _ NULL _ POINTER _ LNG | 需要非空指针的程序接受一个空指针 |
| 4 | LSERR _ INVALID _ INPUT _ LNG | 输入参数不合理 |
| 5 | LSERR _ INFO _ NOT _ AVAILABLE _ LNG | 关于信息的请求不能得到满足 |

## 11.1.5 使用 LINGO DLL 和 Visual C++求解员工安排模型

在本节中,将说明如何应用 Microsoft Visual C/C++建立与 LINGO DLL 接口的程序,以求解员工安排模型。阅读这部分内容之前,需要读者非常熟悉 Visual C++。

下面介绍编写 C++程序求解员工安排模型的步骤。如果想跳过编写程序的细节而直接验证程序的结果,可以在路径 \LINGO8\DLL\VC++\STAFF1\STAFF.EXE 中找到此程序。

首先利用 Visual C++的 AppWizard 工具生成所需的代码,这些代码形成一个基于对话框的应用程序。也就是说,应用程序的界面仅由一个对话框组成。对话框将被修改成类似于下文描述的形式,其中包含员工需求数、每天开始工作的员工数等几项内容。

本程序的源代码可在 \DLL\VC++\STAFF1 子目录下找到,或者用户可以使用 Visual C/C++ 6.0 建立自己的程序源代码,操作步骤如下:

1) 启动 Visual C++,运行 File|New 命令;

2) 选择 Project 页,给程序命名,选择 MFC AppWizard (Exe) 作为应用程序的类型,点击 OK 按钮;

3) 点击 Dialog based 单选按钮,然后点击 Next 按钮;

4) 不选 About box 复选框,因为 LINGO DLL 中的一些功能需要 OLE 的支持,所以选择 OLE Automation 复选框和 OLE Controls 复选框,然后点击 Next 按钮;

5) 再次点击 Next 按钮;

6) 点击 Finish 按钮。

完成以上步骤后,会出现如图 11-3 所示的程序梗概内容。

点击 OK 按钮,AppWizard 将产生框架代码。这时,可以使用源码编辑器将应用程序的对话框修改为图 11-4 所示形式。

接下来,用户需要在 Visual C++中使用 ClassWizard,将成员变量和对话框中的每个部分联系起来。在 View 菜单中,选择 ClassWizard 命令,然后选择 Member Variables 页。

然后,通过运行 Project|Add to Project|Files 命令,将 \LINGO8\DLL\LINGD80.LIB 添加到程序中,以便代码可以使用 LINGO DLL。

做完这一步,就已经将 LINGO DLL 的定义添加到程序中,也就是在对话框中类的头文件顶部添加了 LINGD80.h 头文件 (代码变化以黑体字列出)。程序如下:

```
// staffDlg.h :header file
//
```

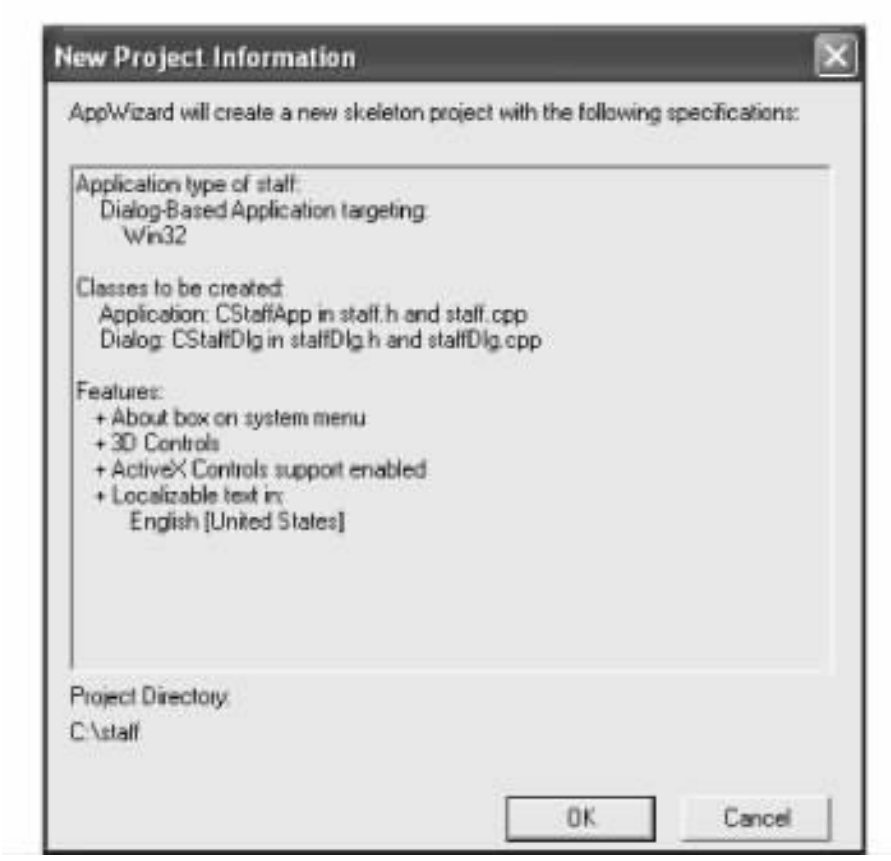



**图** 11-3　**程序梗概内容**

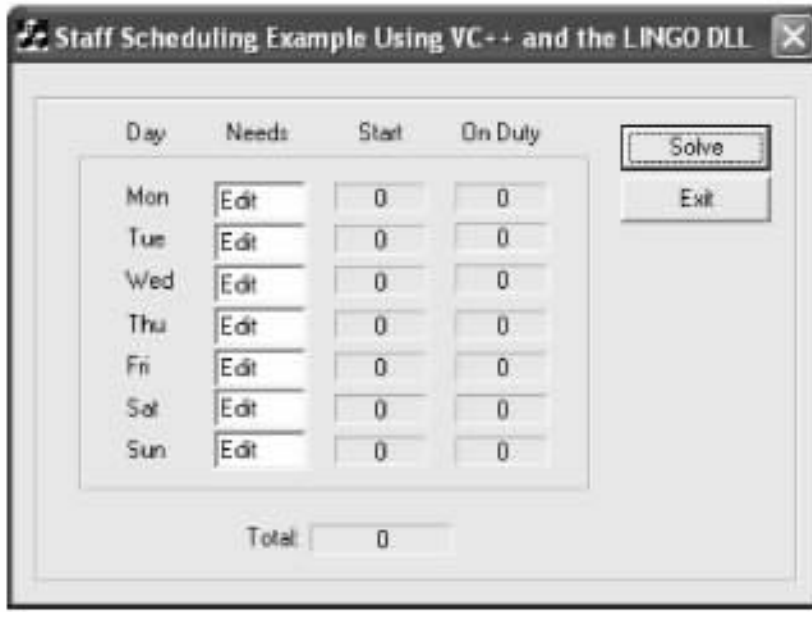



**图** 11-4　Visual C＋＋**程序界面**

```
#include "lingd80.h"
#if
!defined(AFX_STAFFDLG_H__74D746B7_CA4D_11D6_AC89_00010240D2AE__INCLUDED_)
#define
AFX_STAFFDLG_H__74D746B7_CA4D_11D6_AC89_00010240D2AE__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
////////////////////////////////////////////////////////////////////////////
// CStaffDlg dialog
:
```

**这时,所有的基础工作已经做完,可以开始编写实质性的代码以调用 LINGO 求解员工安排模型。转到程序的 Resource View 部分,打开对话框源码,双击 Solve 按钮,会被提示建立按钮名为 OnSolve 的处理函数,这时就可以编辑 OnSolve 的基本版本。它包含如下的代码:**

```
:
void CStaffDlg::OnSolve()
{
  int nError, nPointersNow;
  CString csScript, cs;
  double dNeeds[7], dStart[7], dOnDuty[7], dStatus, dTotal;
  // 从对话框中获得每天所需员工数量
  UpdateData();
  // 将员工需求信息载入数组以便传递给 LINGO
  // LINGO 将所有变量按双精度型处理
  dNeeds[0] = (double) m_nNeedsMon;
  dNeeds[1] = (double) m_nNeedsTue;
  dNeeds[2] = (double) m_nNeedsWed;
  dNeeds[3] = (double) m_nNeedsThu;
  dNeeds[4] = (double) m_nNeedsFri;
  dNeeds[5] = (double) m_nNeedsSat;
  dNeeds[6] = (double) m_nNeedsSun;
  // 创建 LINGO 环境对象
  pLSenvLINGO pLINGO;
  pLINGO = LScreateEnvLng();
  if (!pLINGO)
  {
    AfxMessageBox("Unable to create LINGO Environment");
    return;
  }
  // 打开 LINGO 日志文件
  nError = LSopenLogFileLng(pLINGO, "h h LINGO8 h h LINGO.log");
```

```
if (nError) goto ErrorExit;
// 将内存指针传递给 LINGO
// @POINTER(1)
nError = LSsetPointerLng(pLINGO, dNeeds, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(2)
nError = LSsetPointerLng(pLINGO, dStart, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(3)
nError = LSsetPointerLng(pLINGO, dOnDuty, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(4)
nError = LSsetPointerLng(pLINGO, &dTotal, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(5)
nError = LSsetPointerLng(pLINGO, &dStatus, &nPointersNow);
if (nError) goto ErrorExit;
// 创建 LINGO 命令脚本
csScript = "SET ECHOIN 1 \n";
csScript = csScript +
"TAKE \\LINGO8\\SAMPLES\\STAFFPTR.LNG \n";
csScript = csScript +
"GO \n";
csScript = csScript +
"QUIT \n";
// 执行脚本
dStatus = -1.e0;
nError = LSexecuteScriptLng(pLINGO, (LPCTSTR) csScript);
// 关闭日志文件
LScloseLogFileLng(pLINGO);
// Any problems?
if (nError || dStatus)
{
  // 出现问题
  AfxMessageBox("Unable to solve!");
} else {
  // 无任何问题则在窗口中显示返回结果
  m_csStartMon.Format("%d", (int) dStart[0]);
  m_csStartTue.Format("%d", (int) dStart[1]);
  m_csStartWed.Format("%d", (int) dStart[2]);
  m_csStartThu.Format("%d", (int) dStart[3]);
  m_csStartFri.Format("%d", (int) dStart[4]);
```

```
        m_csStartSat.Format("%d", (int) dStart[5]);
        m_csStartSun.Format("%d", (int) dStart[6]);
        m_csOnDutyMon.Format("%d", (int) dOnDuty[0]);
        m_csOnDutyTue.Format("%d", (int) dOnDuty[1]);
        m_csOnDutyWed.Format("%d", (int) dOnDuty[2]);
        m_csOnDutyThu.Format("%d", (int) dOnDuty[3]);
        m_csOnDutyFri.Format("%d", (int) dOnDuty[4]);
        m_csOnDutySat.Format("%d", (int) dOnDuty[5]);
        m_csOnDutySun.Format("%d", (int) dOnDuty[6]);
        m_csCost.Format("%g", dTotal);
        UpdateData(FALSE);
      }
      goto Exit;
    ErrorExit:
      cs.Format("LINGO Errorcode:%d", nError);
      AfxMessageBox(cs);
      return;
    Exit:
      LSdeleteEnvLng(pLINGO);
    }
```

**注意 如果程序运行中出现错误(若 LINGO 按默认路径安装一般不会出现此问题),有可能是由于在相对路径中找不到程序文件。上述程序中共有两处文件名称及相对路径,分别为日志文件和 LINGO 模型文件,用绝对路径或者把模型文件放到程序文件的同一目录下可以解决此问题。以下内容将对程序的各个部分作详细的解释。**

OnSolve **的第一部分直接从对话框中提取用户关于员工需求的信息。需要注意的是,数据是以双精度型而不是以整型存储的,因为这些值要输入到** LINGO **中,而只有双精度型才能被传入。**

**使用如下代码实现在首次调用** LINGO **时创建** LINGO **环境对象:**

```
// 创建 LINGO 环境对象
pLSenvLINGO pLINGO;
pLINGO = LScreateEnvLng();
if (!pLINGO)
 {
   AfxMessageBox("Unable to create LINGO Environment");
   return;
 }
```

pLSenvLINGO **数据类型定义在** LINGO **的头文件** LINGD80.h **中。**

**然后,建立如下访问** LINGO **的日志文件:**

```
// 打开 LINGO 日志文件
```

```
nError = LSopenLogFileLng(pLINGO, "hhLINGO8\hhLINGO.log");
if (nError) goto ErrorExit;
```

正如上文提到的,当调试程序时,打开 LINGO 日志文件非常有用。如果有些操作发生了错误,日志文件通常能提供有用的线索。

为了确定模型的数据域中使用的 @POINTER() 函数的指向,下一步需要把内存的物理地址传送给 LINGO:

```
// 将指针传递给 LINGO
// @POINTER(1)
nError = LSsetPointerLng(pLINGO, dNeeds, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(2)
nError = LSsetPointerLng(pLINGO, dStart, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(3)
nError = LSsetPointerLng(pLINGO, dOnDuty, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(4)
nError = LSsetPointerLng(pLINGO, &dTotal, &nPointersNow);
if (nError) goto ErrorExit;
// @POINTER(5)
nError = LSsetPointerLng(pLINGO, &dStatus, &nPointersNow);
if (nError) goto ErrorExit;
```

简单地说,当调用 LINGO 求解模型时,@POINTER(1)把员工需求数传送到 LINGO 的 dNeeds 数组。求解信息分别通过@POINTER(2)至@POINTER(5)从 LINGO 中传回到应用软件的 dStart、dOnDuty、dTotal 和 dStatus 等数据结构中。

接下来,用以下代码构建命令脚本:

```
// 创建 LINGO 命令脚本
csScript = "SET ECHOIN 1 h n";
csScript = csScript +
  "TAKE hhLINGO8\hhSAMPLES\hhSTAFFPTR.LNG h n";
csScript = csScript +
  "GO h n";
csScript = csScript +
  "QUIT h n";
```

这个脚本包括了四个命令,每个命令以换行符(ASCII 10)结束,整个脚本以 NULL(ASCII 0)结束。下面的语句是在 LINGO 中执行命令脚本:

```
// 执行脚本
  dStatus = -1.e0;
  nError = LSexecuteScriptLng(pLINGO, (LPCTSTR) csScript);
```

注意：dStatus 被初始化为 - 1。LINGO 通过内存位置号 5 (也就是 @POINTER (5)) 将模型的状态传送到变量 dStatus 中。如果可以求解这个模型，LINGO 将返回一个状态值。如果有意外的错误发生，LINGO 不会到达解出结果的状态。这样，dStatus 的值也不会被改变。所以，将 dStatus 初始化为一个负值，可以检验上述错误情况。假设正常状态下 LINGO 只返回非负状态值，则负的代码值代表出现了问题。这种显示错误的方法是有效的，但并不是最好的。下面将介绍通过指定一个错误回调程序来显示错误的方法。

通过调用 LScloseLogFileLng ()，可以关闭 LINGO 的日志文件，语句如下：

```
// 关闭日志文件
LScloseLogFileLng (pLINGO);
```

接下来，利用下面代码检验 LINGO 是否能够找到一个最优解，语句如下：

```
// 出现问题
if (nError || dStatus)
   // Had a problem
   AfxMessageBox ("Unable to solve !");
} else {
```

注意 在 nError 中的返回代码仅适合于脚本处理器的运行错误，与模型的求解状态没有关系。因此，模型的求解状态需要通过应用 @POINTER () 和 @STATUS () 函数由 LINGO 返回到 dStatus 中。也就是说，一个模型可能有不可行解或者无界解，但是通过 LSexecuteScript () 返回的错误代码将不会给出提示。因此，有必要在程序中添加一个功能用于将模型的求解状态返回到调用程序中。在本程序中，通过使用链接 @STATUS () — @POINTER (5) — dStatus 来实现这一功能。如果发生任何一种错误，最终的模型解将显示为“Unable to solve”。当然，用户界面更为友好的软件应该根据具体的错误情况提供特定信息。

为了避免程序始终占用内存，最后需要注意在程序运行结束后释放 LINGO 环境对象，语句如下：

```
Exit:
   LSdeleteEnvLng (pLINGO);
```

如果正确输入了以上所有的内容，用户就可以建立和运行该 Visual C + + 程序了。

### 11.1.6 使用 LINGO DLL 和 Visual Basic 求解员工安排模型

在这一部分内容中，说明怎样应用 Microsoft Visual Basic 构建应用程序和 LINGO DLL 的接口来求解员工安排问题。在阅读这一部分内容之前，需要读者对 Visual Basic 进行学习。如果读者想跳过构建程序的细节并直接验证完成的应用程序，可以在下列路径下找到该程序：h LINGO8 h DLL h VBASIC h STAFF1 h STAFFVB. EXE。当然，用户也可以按照下列步骤建立此程序：

1) 运行 Visual Basic，点击 File | New Project 命令；

2) 选择 Standard Exe，点击 OK 按钮；

3) 使用资源编辑器，编辑程序的界面如图 11-5 所示。

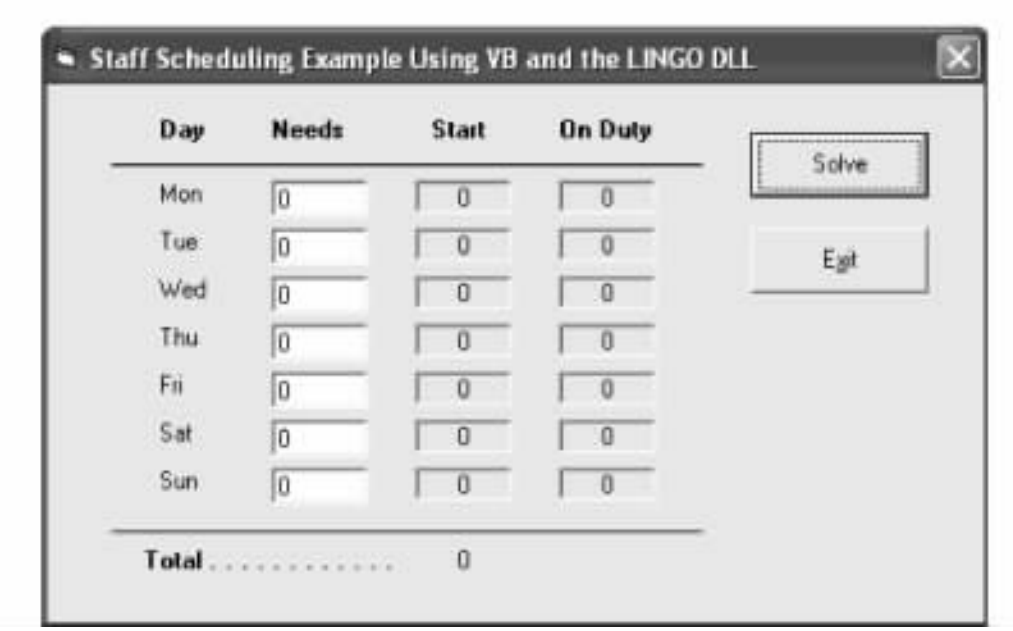



图 11-5　Visual Basic 程序界面

这时，可以为窗口上的两个按钮添加代码。Exit 按钮的功能比较简单，实现在用户单击 Exit 按钮时退出应用软件。双击 Exit 按钮，输入如下代码：

```
Private Sub Exit_Click()
  End
End Sub
```

实现 Solve 按钮的功能需要较多的工作。单击 Solve 按钮时，应用程序会从窗口中读取员工的需求数，并将它们和模型一起传送到 LINGO 的脚本处理器（LGVBSCRIPT）中进行模型求解，最后返回结果并将结果显示在程序对话框中。

在为 Solve 按钮添加代码之前，必须声明 LINGO 的外部函数。通过使用 Visual Basic 中的 Project|Add 模块命令，将 h LINGO8 h DLL h LINGD80.BAS 模块添加到程序中。这个模块包括 LINGO DLL 中所有外部函数的定义。这样，这些函数就可以在程序中应用。在程序对话框中，双击 Solve 按钮，输入如下代码：

```
Private Sub Solve_Click()
'调用 LINGO DLL 求解员工安排模型；
'模型文件为 STAFFPTR.LNG；
'从程序的对话框中获得输入数据；
'从对话框中获得每天所需员工数目；
  Dim dNeeds(7) As Double
  For i = 1 To 7
    dNeeds(i) = Needs(i - 1).Text
  Next i
'创建 LINGO 环境对象
```

```
    Dim pLINGO As Long
    pLINGO = LScreateEnvLng()
    If pLINGO = 0 Then
      MsgBox ("Unable to create LINGO Environment.")
      GoTo FinalExit
    End If
'打开 LINGO 日志文件
    Dim nError As Long
    nError = LSopenLogFileLng(pLINGO, "h:\LINGO8\h\LINGO.log")
    If nError <> 0 Then GoTo ErrorExit
'将指针传递给 LINGO
    Dim dStart(7) As Double, dOnDuty(7) As Double
    Dim dTotal As Double, dStatus As Double
'  @POINTER(1)
    nError = LSsetPointerLng(pLINGO, dNeeds(1), nPointersNow)
    If nError <> 0 Then GoTo ErrorExit
'  @POINTER(2)
    nError = LSsetPointerLng(pLINGO, dStart(1), nPointersNow)
    If nError <> 0 Then GoTo ErrorExit
'  @POINTER(3)
    nError = LSsetPointerLng(pLINGO, dOnDuty(1), nPointersNow)
    If nError <> 0 Then GoTo ErrorExit
'  @POINTER(4)
    nError = LSsetPointerLng(pLINGO, dTotal, nPointersNow)
    If nError <> 0 Then GoTo ErrorExit
'  @POINTER(5)
    nError = LSsetPointerLng(pLINGO, dStatus, nPointersNow)
    If nError <> 0 Then GoTo ErrorExit
'创建 LINGO 命令脚本(每个命令终止使用换行符 ASCII 10)
    Dim cScript As String
'提示命令行开始
    cScript = "SET ECHOIN 1" & Chr(10)
'载入 LINGO 模型
    cScript = cScript & _
      "TAKE  h:\LINGO8\h\SAMPLES\h\STAFFPTR.LNG" & Chr(10)
'模型求解
    cScript = cScript & "GO" & Chr(10)
'退出 LINGO DLL
    cScript = cScript & "QUIT" & Chr(10)
'使用 NULL 符(ASCII 0)终止脚本
    cScript = cScript & Chr(0)
'执行脚本
```

```
    dStatus = - 1 #
    nError = LSexecuteScriptLng (pLINGO, cScript)
'关闭日志文件
    LScloseLogFileLng (pLINGO)
'出现问题
    If nError > 0 Or dStatus < > 0 Then
      MsgBox ("Unable to solve !")
      GoTo ErrorExit
    End If
'返回每天开始工作员工数目
    For i = 1 To 7
      Start (i - 1) . Caption = dStart (i)
    Next i
'返回每天当值员工数目
    For i = 1 To 7
      OnDuty (i - 1) . Caption = dOnDuty (i)
    Next i
'返回一周雇用员工总数
    Total . Caption = dTotal
    LSdeleteEnvLng (pLINGO)
    GoTo FinalExit :
ErrorExit :
    MsgBox ("LINGO Error Code :" & nError&)
    LSdeleteEnvLng (pLINGO)
FinalExit :
End Sub
```

Solve _ Click 过程中的第一部分比较简单,实现从对话框中直接提取员工需求数。同样,数据的存储格式是双精度型的。下面对代码的每一部分进行解释。

使用如下代码实现第一次调用 LINGO 时创建 LINGO 环境对象:

```
'创建 LINGO 环境对象
    Dim pLINGO As Long
    pLINGO = LScreateEnvLng ()
    If pLINGO = 0 Then
      MsgBox ("Unable to create LINGO Environment.")
      GoTo FinalExit
    End If
```

接下来,调用 LINGO 的日志文件:

```
'打开 LINGO 日志文件
    Dim nError As Long
    nError = LSopenLogFileLng (pLINGO, " h LINGO8 h LINGO. log")
```

```
    If nError < > 0 Then GoTo ErrorExit
```

如上所述,使用 LINGO 的日志文件对于调试程序非常有用。如果出现一些错误,日志文件常会提供有帮助的线索。

为了确定模型的数据域中使用的 @POINTER () 函数的指向,需要把内存的物理地址传送给 LINGO。以下步骤完成这一操作:

```
'将指针传递给 LINGO
  Dim dStart (7) As Double, dOnDuty (7) As Double
  Dim dTotal As Double, dStatus As Double
' @POINTER (1)
  nError = LSsetPointerLng (pLINGO, dNeeds (1) , nPointersNow)
  If nError < > 0 Then GoTo ErrorExit
' @POINTER (2)
  nError = LSsetPointerLng (pLINGO, dStart (1) , nPointersNow)
  If nError < > 0 Then GoTo ErrorExit
' @POINTER (3)
  nError = LSsetPointerLng (pLINGO, dOnDuty (1) , nPointersNow)
  If nError < > 0 Then GoTo ErrorExit
' @POINTER (4)
  nError = LSsetPointerLng (pLINGO, dTotal, nPointersNow)
  If nError < > 0 Then GoTo ErrorExit
' @POINTER (5)
  nError = LSsetPointerLng (pLINGO, dStatus, nPointersNow)
  If nError < > 0 Then GoTo ErrorExit
```

简单地说,当调用 LINGO 求解模型时,@POINTER (1) 把员工需求传送到 LINGO 的 dNeeds 列。求解信息分别通过 @POINTER (2) 至 @POINTER (5) 从 LINGO 中传回到应用程序的 dStart、dOnDuty、dTotal 和 dStatus 等数据结构中。

以下代码用来构建命令脚本:

```
'创建 LINGO 命令脚本 (每个命令终止使用换行符 ASCII 10)
  Dim cScript As String
' Causes LINGO to echo input'提示命令行开始
  cScript = "SET ECHOIN 1" & Chr (10)
'载入 LINGO 模型
  cScript = cScript & _
  "TAKE  h LINGO8 h SAMPLES h STAFFPTR. LNG" & Chr (10)
'模型求解
  cScript = cScript & "GO" & Chr (10)
'退出 LINGO DLL
  cScript = cScript & "QUIT" & Chr (10)
'使用 NULL 符 (ASCII 0) 终止脚本
```

```
cScript = cScript & Chr(0)
```

下面语句用于在 LINGO 中执行命令脚本：

```
'执行脚本
dStatus = -1#
nError = LSexecuteScriptLng(pLINGO, cScript)
```

dStatus 的用法参见 Visual C++ 部分。现在，调用 LScloseLogFileLng() 关闭 LINGO 的日志文件：

```
'关闭日志文件
LScloseLogFileLng (pLINGO)
```

以下代码用来测试 LINGO 是否能够找到最优解：

```
'出现问题
If nError > 0 Or dStatus < > 0 Then
  MsgBox ("Unable to solve !")
  GoTo ErrorExit
End If
```

和 Visual C++ 中的程序一样，在 nError 中的返回代码仅适合于脚本处理器的运行错误，与模型的求解状态没有关系。因此，需要 nError 和 dStatus 两个变量相互配合，发现程序运行和模型求解中的所有问题。

为了避免程序始终占用内存，最后需要注意在程序结束运行后释放 LINGO 环境对象：

```
LSdeleteEnvLng (pLINGO)
GoTo FinalExit：
```

如果已经正确输入以上所有的内容，用户就可以建立和运行该 Visual Basic 程序了。

## 11.1.7 使用 LINGO DLL 和 Delphi 求解员工安排模型

在这部分内容中，将介绍如何应用 Delphi 建立与 LINGO DLL 的接口程序来求解员工安排模型。同样，需要读者比较熟悉 Delphi 的使用。

首先，按照如下步骤建立该程序：

1) 运行 Delphi (本书以 Delphi 5.0 为例)，点击 File|New 命令；

2) 选择 Application，点击 OK 按钮；

3) 使用对象编辑器，编辑程序的界面如图 11-6 所示。

然后，为窗口上的两个按钮添加代码。Exit 按钮的功能比较简单，用户单击 Exit 按钮时退出应用程序。在 Delphi 编程环境中双击 Exit 按钮，输入如下代码：

```
procedure TForm1.btn_ExitClick(Sender：TObject)；
begin
  Close；
end；
```

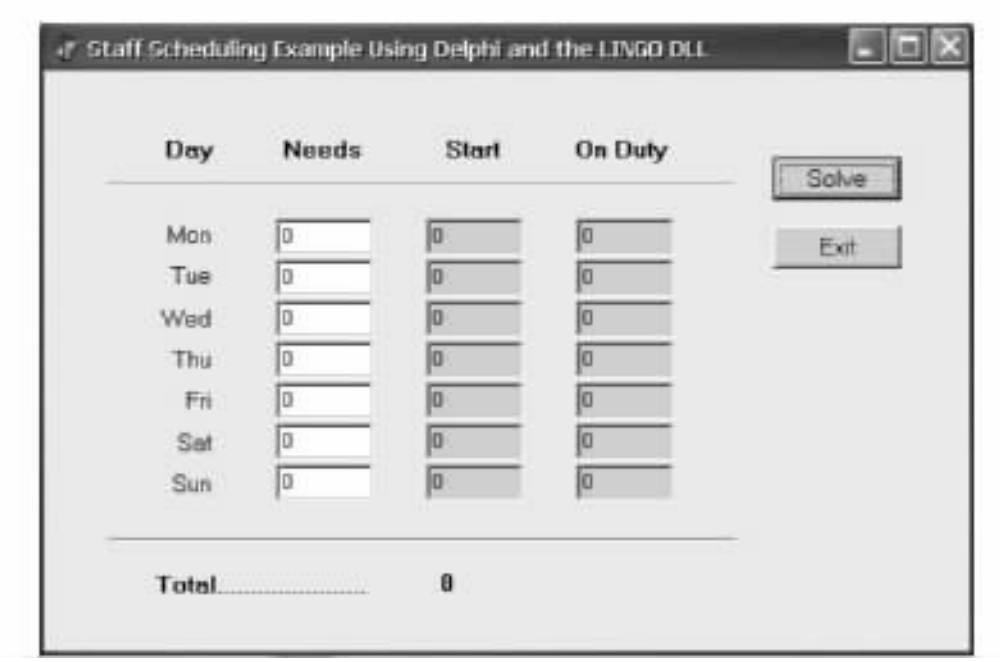



图 11-6　Delphi 程序界面

实现 Solve 按钮的功能则需要较多的工作。用户单击 Solve 按钮时，应用程序会从窗口中读取员工的需求数，将它们和模型一起传送到 LINGO 的脚本处理器中进行模型求解，最后返回结果并将其显示在对话框中。

在编写 Solve 按钮的代码之前，必须调用 LINGO 的外部函数单元。首先通过使用 Delphi 中的 Project | Add to Project . . . 命令，将 h LINGO8 h DLL h Lingodel . pas 单元添加到程序中。这个单元包括 LINGO DLL 中所有外部函数的定义。这样，这些函数就可以在程序中使用。为了完成调用，还需要在程序中引用刚刚添加的外部函数单元，可以在相应位置中输入如下引用代码：

```
implementation
uses lingodel;                                    //引用外部函数单元
{$ R * . DFM}
```

这时，就可以添加 Solve 按钮的代码了。在 Delphi 编译环境下，双击 Solve 按钮，输入如下代码：

```
procedure TForm1 . btn _ SolveClick (Sender : TObject) ;
label FinalExit, NormalExit, ErrorExit;
var
  dNeeds, dStart, dOnDuty : array [1 . . 7] of Double;
  dTotal, dStatus : Double;
  nError, pLingo, nPointersNow : integer;
  cScript : pchar;
  cMsg : string;
  i : integer;
```

```
begin
  //在本例中,模型文件 STAFFPTR.LNG 需要与 Delphi 执行文件置于同目录下
  //从程序的对话框中获得输入数据
  for i := 1 to 7 do
    dNeeds[i] := StrToFloat(TEdit(FindComponent('Needs' + inttostr(i))).Text);
  //创建 LINGO 环境对象
  pLingo := LScreateEnvLng();
  if (pLingo = 0) then
  begin
    ShowMessage('Can''t create LINGO environment');
    goto FinalExit;
  end;
  //打开 LINGO 日志文件
      nError := LSopenLogFileLng(pLINGO, 'LINGO.log');
      if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //将指针传递给 LINGO
  //@POINTER(1)
  nError := LSsetPointerLng(pLINGO, dNeeds[1], nPointersNow);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //@POINTER(2)
  nError := LSsetPointerLng(pLINGO, dStart[1], nPointersNow);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //@POINTER(3)
  nError := LSsetPointerLng(pLINGO, dOnDuty[1], nPointersNow);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //@POINTER(4)
  nError := LSsetPointerLng(pLINGO, dTotal, nPointersNow);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //@POINTER(5)
  nError := LSsetPointerLng(pLINGO, dStatus, nPointersNow);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //创建 LINGO 命令脚本
  cScript := 'SET ECHOIN 1' + Char(10) +
    'TAKE STAFFPTR.LNG' + Char(10) +
    'GO' + Char(10) +
    'QUIT' + Char(10) +
    Char(0);
   //执行脚本
  nError := LSexecuteScriptLng(pLINGO, cScript);
  if (nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
  //关闭日志文件
  LScloseLogFileLng(pLINGO);
```

```
//出现问题
if ((nError < > LSERR _ NO _ ERROR _ LNG) or
   (dStatus < > LS _ STATUS _ GLOBAL _ LNG)) then
   ShowMessage ('Unable to solve !')
//无任何问题则在窗口中显示返回的结果
else
begin
   //返回每天开始工作员工数目
   for i := 1 to 7 do
      TEdit (FindComponent ('Start' + inttostr (i))). Text := FloatToStr (dStart [i]);
   //返回每天当值员工数目
   for i := 1 to 7 do
      TEdit (FindComponent ('OnDuty' + inttostr (i))). Text := FloatToStr (dOnDuty [i]);
   //返回一周雇用员工总数
   Total. Caption := FloatToStr (dTotal);
end;
goto NormalExit;
ErrorExit:
   cMsg := Format ('LINGO Errorcode: %d', [nError]);
   ShowMessage (cMsg);
   goto FinalExit;
NormalExit:
   //释放 LINGO 环境对象以避免占用内存
   LSdeleteEnvLng (pLINGO);
FinalExit:
end;
```

Solve 按钮代码的第一部分比较简单,完成从对话框中直接提取每天需求的员工数目。同样,数据的存储格式是双精度型的,而不是整数型。下面对代码的每一部分进行解释。

实现第一次调用 LINGO 时创建 LINGO 环境对象使用如下代码:

```
//创建 LINGO 环境对象
pLingo := LScreateEnvLng ();
if (pLingo = 0) then
begin
   ShowMessage ('Can''t create LINGO environment');
   goto FinalExit;
end;
```

接下来,调用 LINGO 的日志文件:

```
//打开 LINGO 日志文件
nError := LSopenLogFileLng (pLINGO, 'LINGO. log');
if (nError < > LSERR _ NO _ ERROR _ LNG) then goto ErrorExit;
```

如上所述，日志文件对于调试程序很有帮助。

为了确定模型数据域中使用的@POINTER()函数的指向，需要把内存物理地址传送给LINGO。以下语句完成这一操作：

```
//将指针传递给 LINGO
//@POINTER(1)
nError := LSsetPointerLng(pLINGO, dNeeds[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(2)
nError := LSsetPointerLng(pLINGO, dStart[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(3)
nError := LSsetPointerLng(pLINGO, dOnDuty[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(4)
nError := LSsetPointerLng(pLINGO, dTotal, nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(5)
nError := LSsetPointerLng(pLINGO, dStatus, nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
```

简单地说，当调用LINGO求解模型时，使用指针@POINTER(1)把员工需求数传送到LINGO的dNeeds列，并且分别使用指针@POINTER(2)到@POINTER(5)把求解结果从LINGO中传回到应用程序的dStart、dOnDuty、dTotal和dStatus等数据结构中。

接下来，使用如下代码建立命令脚本：

```
//创建 LINGO 命令脚本
cScript := 'SET ECHOIN 1' + Char(10) +
  'TAKE STAFFPTR.LNG' + Char(10) +
  'GO' + Char(10) +
  'QUIT' + Char(10) +
  Char(0);
```

下面语句用于在LINGO中执行命令脚本：

```
//执行脚本
nError := LSexecuteScriptLng(pLINGO, cScript);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
```

现在，调用LScloseLogFileLng()关闭LINGO的日志文件：

```
//关闭日志文件
LScloseLogFileLng(pLINGO);
```

然后，利用如下代码测试LINGO是否能够找到最优解：

```
//出现问题
```

```
if ((nError < > LSERR _ NO _ ERROR _ LNG) or
(dStatus < > LS _ STATUS _ GLOBAL _ LNG)) then
  ShowMessage ('Unable to solve !') ;
```

和 Visual C + + 以及 Visual Basic 中的程序一样，在 nError 中的返回代码仅适合于脚本处理器的运行错误，与模型的求解状态没有关系。因此，需要 nError 和 dStatus 两个变量相互配合，以便发现程序运行和模型求解中的所有问题。

如果没有任何问题，程序窗口中将显示返回的计算结果，代码如下：

```
//无任何问题则在窗口中显示返回结果
else
begin
  //返回每天开始工作员工数目
  for i := 1 to 7 do
  TEdit (FindComponent ('Start' + inttostr (i))) . Text := floatToStr (dStart [i]) ;
  //返回每天当值员工数目
  for i := 1 to 7 do
  TEdit (FindComponent ('OnDuty' + inttostr (i))) . Text := floatToStr (dOnDuty [i]) ;
  //返回一周雇用员工总数
  Total . Caption := floatToStr (dTotal) ;
end ;
```

在程序运行结束后要释放 LINGO 环境对象，代码如下所示：

```
//释放 LINGO 环境对象以避免占用内存
LSdeleteEnvLng (pLINGO) ;
```

如果已经正确输入上述所有内容，用户就可以建立和运行该 Delphi 程序了。

## 11.1.8 回调函数

在许多情况下，求解模型耗时很长，因此，帮助用户了解最优化进程的信息就显得很重要。LINGO 标准交互式版本通过一个状态窗口显示每次调用求解器时模型的迭代次数、目标值和其他各种模型统计信息。此外，用户还可以通过单击状态窗口上的一个按钮中断求解过程。通过使用 LINGO DLL，可以为用户提供类似的功能。为了达到这些目的，需要向 LINGO 提供一个回调函数。

在下面的章节中，将介绍回调函数中调用序列的应用。然后，以 Visual C + + 和 Visual Basic 的员工安排模型为例，说明回调函数的使用方法。

### 11.1.8.1 指定回调函数

为了指定回调函数，调用 LINGO 的脚本处理器之前需要首先调用 LINGO 的输出程序 LSsetCallback。这样回调函数就可以被 LINGO 求解器调用。LSsetCallbackSolverLng 的调用序列如下：

```
int LSsetCallbackSolverLng (pLSenvLINGO pL, lngCBFuncError _ t pcbf, void * pUserData)
```

参数意义：

pL——由 LScreateEnvLng() 函数创建的指向 LINGO 环境的指针；

pcbf——指向回调函数的指针；

pUserData——用户的特定指针，用户可以用它指向回调函数中需要参阅的任何数据，LINGO 只通过回调函数传送指针指向的数据，如果不需要这个指针，设置指针值为 NULL。

返回值：

如果没有发生错误返回 0，否则将返回一个非零错误代码。

回调函数必须使用标准访问习惯并且具有如下接口：

```
int MySolverCallback(pLSenvLINGO pL, int nReserved, void * pUserData);
```

如果没有严格遵守接口规则，计算机极有可能瘫痪。nReserved 参数可以保留以备后用，也可以被忽略。

如果回调函数被正确使用，有关求解器状态的信息将会通过使用 LSgetCallbackInfoLng() 函数从 LINGO 中传回。

#### 11.1.8.2 Visual C++ 回调函数

现在通过引入回调函数，扩展使用 Visual C++ 求解员工安排模型的例子。在此例中，每当求解器找到一个更好的整数解时，回调函数将传回一个对话框，此对话框用来显示求解器迭代次数、新结果的目标值以及目标的边界，并且含有一个 Interrupt 按钮，允许用户在需要时中断求解器的运行。这个例子的完整程序可以在路径 h LINGO8 h DLL h VC++ h STAFF2 中找到。建立该程序的具体步骤如下。

第一步，设计对话框。每当一个新的整数解求出时，显示该对话框。此例的对话框如图 11-7 所示。该对话框中有三个编辑区域，即迭代次数（Iteration）、目标值（Objective）和边界（Bound）。另外，还有两个按钮，Interrupt 按钮用来中断求解器的运行，OK 按钮用来关闭对话框。

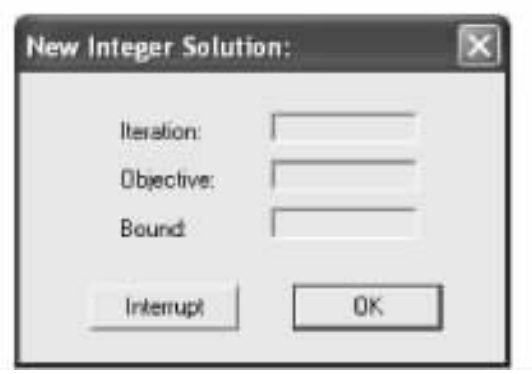



图 11-7 回调函数对话框

接下来，用 ClassWizard 在上述新的对话框上添加一个名为 CNewIPDlg 的类，并指定成员变量以处理迭代次数、目标值和边界的编辑域。完成这些后，对话框的头文件如下：

```
// NewIPDlg.h : header file
//
#include "resource.h"
/////////////////////////////////////////////////////////////////////////
// CNewIPDlg dialog
class CNewIPDlg : public CDialog {
// Construction
```

```
public:
CNewIPDlg(CWnd* pParent = NULL);//standard constructor
// Dialog Data
  //{{AFX_DATA(CNewIPDlg)
  enum {IDD = IDD_NEW_IP_SOLUTION};
  CStringm_csBound;
  CStringm_csIteration;
  CStringm_csObjective;
  //}}AFX_DATA
// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CNewIPDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);
//}}AFX_VIRTUAL
// Implementation
protected:
  // Generated message map functions
//{{AFX_MSG(CNewIPDlg)
// NOTE:the ClassWizard will add member functions here
//}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};
```

**以下是处理对话框中的事件的代码：**

```
// NewIPDlg.h :header file
//
/////////////////////////////////////////////////////////////////////
// CNewIPDlg dialog
class CNewIPDlg :public CDialog
{
// Construction
public:
CNewIPDlg(CWnd* pParent = NULL);   // standard constructor
// Dialog Data
  //{{AFX_DATA(CNewIPDlg)
  enum {IDD = IDD_NEW_IP_SOLUTION};
  CStringm_csBound;
  CStringm_csIteration;
  CStringm_csObjective;
//}}AFX_DATA
// Overrides
```

```
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNewIPDlg) protected:
    virtual void DoDataExchange(CDataExchange* pDX);
  //}}AFX_VIRTUAL
  // Implementation
  protected:
  // Generated message map functions
    //{{AFX_MSG(CNewIPDlg)
      // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

**以上两个文件中的代码全部由** ClassWizard **生成,不需要用户输入。**

**第二步,必须添加回调函数的代码。以下全局程序就是为了实现这个目的:**

```
void* int _stdcall MyCallback(void* pModel, int nReserved, pUserData)
{
// 回调函数
  // 获得当前最优整数解
  int nErr;
  double dBestIP;
  nErr = LSgetCallbackInfoLng(pModel,S_DINFO_MIP_BEST_OBJECTIVE_LNG, &dBestIP);
  if (nErr) return(0);
  // 将最优整数解显示在对话框中
  double* pdBestIPShown = (double*) pUserData;
  // 检验最新的目标值是否比以前的更好
  if (dBestIP < *pdBestIPShown)
  {
      // 若更好则显示结果
      *pdBestIPShown = dBestIP;
      // 从 LINGO 中获取迭代次数
      int nIterations;
      LSgetCallbackInfoLng(pModel, LS_IINFO_ITERATIONS_LNG,
      &nIterations);
      // 从 LINGO 中获取目标值边界
      double dBound;
      LSgetCallbackInfoLng(pModel,LS_DINFO_MIP_BOUND_LNG, &dBound);
      // 创建一个对话框来显示结果
      CNewIPDlg dlgNewIP;
      // 初始化对话框
      dlgNewIP.m_csIteration.Format("%d", nIterations);
      dlgNewIP.m_csObjective.Format("%d", (int) dBestIP);
```

```
            dlgNewIP.m_csBound.Format("%d", (int) dBound);
            // 显示对话框并判断用户是否点击中断按钮
            if (dlgNewIP.DoModal() == IDCANCEL) return(-1);
        }
        return(0);
    }
```

下面对上述程序作详细解释。以下代码是使用 LSgetCallbackInfo 程序获得当前最优整数解的值：

```
int nErr;
double dBestIP;
nErr = LSgetCallbackInfoLng(pModel,
 LS_DINFO_MIP_BEST_OBJECTIVE_LNG, &dBestIP);
if (nErr) return(0);
```

常量 LS_DINFO_MIP_BEST_OBJECTIVE 定义在 LINGD80.H 头文件中。

这样就得到了最优整数解的值，然后使用如下语句将其显示在对话框中：

```
double * pdBestIPShown = (double *)pUserData;
```

注意 这条语句调用用户数据指针 pUserData。当回调函数建立以后，这个指针被传送到 LINGO 中，使用此指针是从回调函数中获取数据的有效方法。在这个特定例子中，指针只指向一个变量。实际上，指针可以指向一个很大的数据结构并且可以包含需要的任何数据。

然后，使用以下语句检验最新得到的整数解是否比其他的解更好：

```
if (dBestIP < *pdBestIPShown)
```

如果新的结果更好，就从 LINGO 中得到迭代次数和结果边界等附加的信息。同时，使用以下语句创建一个回调函数对话框的实例以显示最新的信息：

```
CNewIPDlg dlgNewIP;
```

这时，可以将新的数据加载到对话框相应的域中：

```
dlgNewIP.m_csIteration.Format("%d", nIterations);
dlgNewIP.m_csObjective.Format("%d", (int) dBestIP);
dlgNewIP.m_csBound.Format("%d", (int) dBound);
```

回调函数的最后一步是显示对话框。如果用户点击中断按钮，则返回 -1，表示求解器停止计算，并且返回到目前为止找到的最优解：

```
// 显示对话框并判断用户是否点击中断按钮
if (dlgNewIP.DoModal() == IDCANCEL) return(-1);
```

这时，还需要添加另外一些代码以调用 LSsetCallbackSolverLng()程序，将 LINGO 中的一个指针传送到回调函数中。添加这些代码的理想位置就是在 Solve 按钮的 OnSolve()代码中刚创建完 LINGO 环境之后的地方。以下代码中的黑体字表示改变部分：

```
// 创建 LINGO 环境对象
pLSenvLINGO pLINGO;
pLINGO = LScreateEnvLng();
if (!pLINGO)
{
   AfxMessageBox("Unable to create LINGO Environment");
   return;
}
//从 LINGO 中传送一个指针到回调程序
nError = LSsetCallbackSolverLng(pLINGO, &MyCallback,
 &dBestIPShown);
if (nError) goto ErrorExit;
// 打开 LINGO 日志文件
nError = LSopenLogFileLng(pLINGO, "h h LINGO8 h h LINGO.log");
if (nError) goto ErrorExit;
```

#### 11.1.8.3 Visual Basic 回调函数

这部分内容通过引入回调函数扩展使用 Visual Basic 解决员工安排模型的例子。与 Visual C++相同,在此例中,每当求解器找到一个更好的整数解时,回调函数传回一个对话框。此对话框用来显示求解器的迭代次数、目标值以及目标的边界。本例的完整程序可以在路径 h LINGO8 h DLL h VBasic h STAFF2 中找到。

首先,需要构建一个回调函数并将它放在一个独立的模块文件中。以下是回调函数 Module.bas 文件中的内容:

```
Public Function MySolverCallback(ByVal pModel As Long, _
 ByVal nReserved As Long, ByRef dBestIP As Double) As Long
' 回调函数
   Dim nReturnVal As Long
   nReturnVal = 0
' 获得当前最优整数解
   Dim dObj As Double
   Dim nError As Long
   nError = LSgetCallbackInfoDoubleLng(pModel, _
LS_DINFO_MIP_BEST_OBJECTIVE_LNG, dObj)
' 检验错误
   If (nError = LSERR_NO_ERROR_LNG) Then
' 检验最新的目标值是否比以前的更好
      If (dObj < dBestIP) Then
' 若更好则显示结果
' 保存新的最优目标值
         dBestIP = dObj
```

```
' 从 LINGO 中获取迭代次数
        Dim nIterations As Long
        nResult = LSgetCallbackInfoLongLng(pModel, _
          LS_IINFO_ITERATIONS_LNG, nIterations)
' 从 LINGO 中获取目标值边界
        Dim dBound As Double
        nResult = LSgetCallbackInfoDoubleLng(pModel, _
          LS_DINFO_MIP_BOUND_LNG, dBound)
' 在对话框中显示信息
        Dim nButtonPressed
        Dim cMessage As String
        cMessage = "Objective:" + Str(dBestIP) _
          + Chr(10) + "Bound:" + Str(dBound) _
          + Chr(10) + "Iterations:" + Str(nIterations)
        nButtonPressed = MsgBox(cMessage, vbOKCancel)
        If (nButtonPressed = vbCancel) Then
            nReturnVal = -1
        End If
    End If
  End If
    MySolverCallback = nReturnVal
End Function
```

**注意**:Visual Basic **回调函数必须存在于模块文件**(.bas)**中。如果存在于窗口文件**(.frm)**中,则回调函数将不能使用。同时,回调函数和模块文件必须使用不同的名称。如果回调函数和模块文件名称相同,**Visual Basic AddressOf **函数不能传回回调函数的地址。**

**如前所述,回调程序必须使用如下的调用序列:**

```
int _stdcall MySolverCallback(pLSenvLINGO pL, int nReserved, void* pUserData)
```

**使用** Visual Basic **代码定义一个等价函数:**

```
Public Function MySolverCallback(ByVal pModel As Long, _
ByVal nReserved As Long, ByRef dBestIP As Double)
```

**因为默认情况下** Visual Basic **使用标准调用习惯**(_stdcall),**所以在此不必明确指定它。**

**用户数据指针**(pUserData)**可以用来传递显示在参数** dBestIP **中的当前最优目标值。比较每个新的目标值和当前最优解,如果最新的解有所改进,则显示一个对话框给出新解的信息。**

**在如下代码中,使用** LSgetCallbackInfo **函数获得目前为止找到的最优整数解:**

```
' 获得当前最优整数解
  Dim dObj As Double
  Dim nError As Long
  nError = LSgetCallbackInfoDoubleLng(pModel, _
    LS_DINFO_MIP_BEST_OBJECTIVE_LNG, dObj)
```

在 Visual Basic 的 LINGO 头文件中（LINGD80.BAS），LSgetCallbackInfoLng()函数有两个等价函数，即 LSgetCallbackInfoDoubleLng() 和 LSgetCallbackInfoLongLng()，它们分别是为了从 LINGO 中传送双精度型和长整型数据。因为在 Visual Basic 中不支持 C 语言中的 void 数据类型，所以通过 LSgetCallbackInfoDoubleLng()传回以双精度型表示的目标值。

检验传回目标值过程中可能发生的任何错误。如果没有发生任何错误，检验最新的目标值是否比以前的更好：

```
' 检验错误
  If (nError = LSERR_NO_ERROR_LNG) Then
' 检验最新的目标值是否比以前的更好
      If (dObj < dBestIP) Then
```

如果新的目标值比以前的更好，保存新的目标值，传回迭代次数和目标值边界：

```
' 保存新的最优目标值
          dBestIP = dObj
' 从 LINGO 中获取迭代次数
          Dim nIterations As Long
          nResult = LSgetCallbackInfoLongLng(pModel, _
            LS_IINFO_ITERATIONS_LNG, nIterations)
' 从 LINGO 中获取目标值边界
          Dim dBound As Double
          nResult = LSgetCallbackInfoDoubleLng(pModel, _
            LS_DINFO_MIP_BOUND_LNG, dBound)
```

接下来，将新结果的概要信息显示在对话框中：

```
' 在对话框中显示信息
          Dim nButtonPressed
          Dim cMessage As String
          cMessage = "Objective:" + Str(dBestIP) _
            + Chr(10) + "Bound:" + Str(dBound) _
            + Chr(10) + "Iterations:" + Str(nIterations)
          nButtonPressed = MsgBox(cMessage, vbOKCancel)
```

如果单击 Cancel 按钮，将中断 LINGO 求解器，并在返回之前将返回值设置为 - 1：

```
          If (nButtonPressed = vbCancel) Then
              nReturnVal = - 1
          End If
      End If
  End If
  MySolverCallback = nReturnVal
End Function
```

这时，还需要添加另外的一些代码，用于调用 LSsetCallbackSolverLng()函数，以将一个 LIN-

GO 中的指针传递到回调函数中。使用如下调用语句在 Solve 按钮处理程序的开始处完成指针的传递：

```
' 从 LINGO 中传递一个指针到回调函数
Dim nError As Long
Dim dBestObj As Double
dBestObj = 1E + 30
nError = LSsetCallbackSolverLng(pLINGO, _
  AddressOf MySolverCallback, dBestObj)
```

这里应注意在调用 LSsetCallbackSolverLng() 函数时 AddressOf 操作的使用。这一操作仅在函数调用中使用，用于传递包含在模块文件中程序的地址。

注意：AddressOf 操作已经添加到从 Visual Basic 5.0 开始的版本中。因此 Visual Basic 早期的版本不能使用 LINGO 中的回调功能。此外，Visual Basic for Application (VBA)、Microsoft Office 提供的 Visual Basic 功能也不支持 AddressOf 操作。因此，VBA 程序访问 LINGO 时不能建立回调函数。

### 11.1.9 总结

LINGO DLL 结构简单，只需要掌握一些用来访问 DLL 的函数以及在应用软件中添加 LINGO 求解功能的操作代码就可以调用它。

本章内容中给出了从 Visual Basic、Visual C + + 以及 Delphi 中调用 DLL 的例子。使用 FORTRAN 和 Java 编写的其他的例子保存在 h LINGO8 h DLL 文件夹中。其他环境下的应用软件的开发者可以使用类似于本章所给例子的方式访问 LINGO DLL。

最后需要注意，建立任何使用 LINGO DLL 的应用程序都受到 LINGO 许可协议的保护。因此，如果没有得到 LINDO 公司许可，不可以发布这样的程序。如果想发布自己的应用程序，请联系 LINDO 公司以获得一定运行时间的许可。

## 11.2 用户定义函数

@USER 函数允许用户在 LINGO 中使用自定义的函数。在 Windows 版本的 LINGO 中，用户可以通过提供 32 位的动态链接库 (DLL) 使用 @USER 函数，而支持 Windows 的大部分程序语言允许用户建立 DLL。在 Windows 以外的系统中，则可以通过向 LINGO 提供 C 或者 FORTRAN 的编译子程序使用 @USER 函数。对于 LINGO 的建模者来说，@USER 函数可以含有任何数量但至少为一个的参数，并且返回用户编写的程序的计算结果。

从编写自定义函数的编程者来看，@USER 函数只考虑两个输入参数，返回一个结果。两个输入参数分别为：

1) 一个整数，指定 LINGO 模型中 @USER 函数的参数的个数；

2) 一个向量，包含 @USER 函数中参数的值，这些值的顺序根据其在 @USER 函数中出现的顺序确定，并且这些值都是双精度型的。

换句话说，虽然对 LINGO 的建模者来说 @USER 函数可以含有任何数量的参数，但是对实

现@USER函数的程序编写者来说,只须传送两个输入参数。

可以通过把每一个函数看作是独立的子程序来编写和编译,这样相当于使用多重@USER函数,并且可以给@USER函数增加一个参数,并将其看作是子程序的索引值。

### 11.2.1 在Windows系统下安装@USER函数

在Windows中启动LINGO时,将搜索名为MYUSER.DLL的DLL文件。LINGO首先在工作目录(保存用户LINGO文件的目录)中搜寻这个文件,如果在工作目录下没有找到,LINGO将搜索启动目录,启动目录是安装LINGO程序的目录。如果LINGO找到了MYUSER.DLL,它将把DLL加载到内存中,每次模型中运行@USER函数时,调用输出的MYUSER程序。

### 11.2.2 Visual C++例子

这一部分内容是用Microsoft Visual C/C++建立一个32位的DLL文件,该文件含有计算平方根的@USER函数。使用Visual C++中的AppWizard可以轻松地建立该DLL文件的基本代码,或者在LINGO主目录下的USER h MSVC子目录中找到这个例子的代码。运行Visual C++ Developer Studio,按如下步骤建立这些基本代码:

1)运行File|New命令;

2)出现一个New对话框,选择Project Workspace选项,点击OK按钮;

3)出现一个New Project Workspace对话框,将工程命名为sqroot,并在Type框中选择MFC AppWizard(dll)选项,然后点击Create按钮;

4)出现一个新的MFC AppWizard对话框,点击Finish按钮;

5)出现New Project Information窗口(图11-8),该窗口包含工程中所选选项的信息概要。

这时,点击OK按钮就完成了DLL基本代码的创建。

现在,编辑sqroot.cpp文件并对其进行修改(黑体字部分):

```
// sqroot.cpp :Defines the initialization
// routines for the DLL.
//
#include "stdafx.h"
#include "sqroot.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////////////////////////
// CSqrootApp
BEGIN_MESSAGE_MAP(CSqrootApp, CWinApp)
//{{AFX_MSG_MAP(CSqrootApp)
        // NOTE:the ClassWizard will add and
            // remove mapping macros here.
```

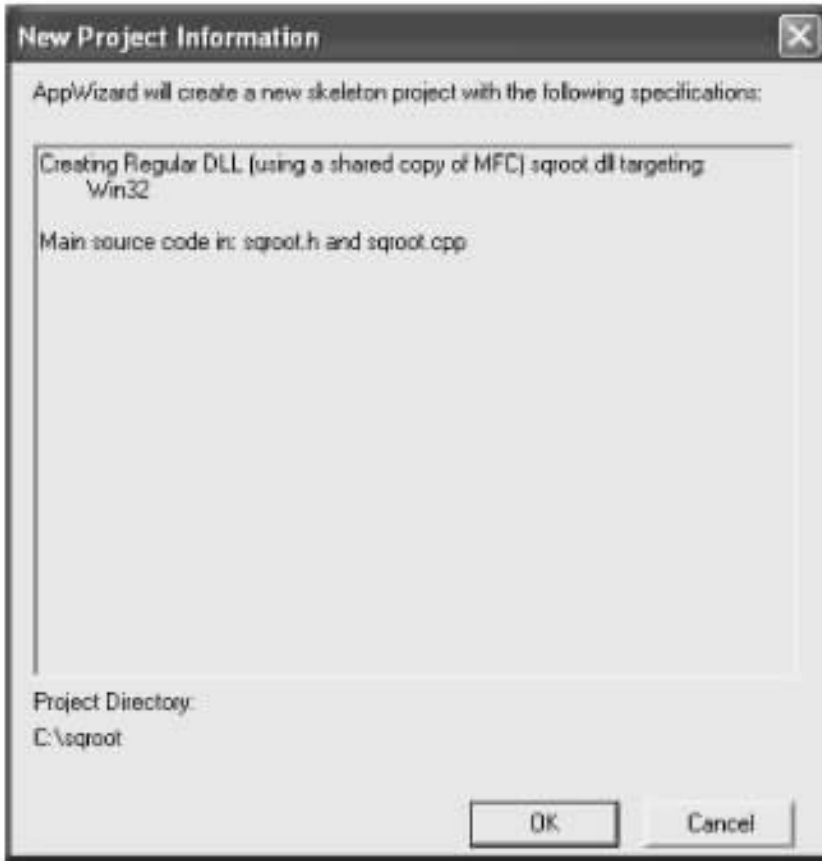



**图** 11-8　**程序梗概内容**

```
        // DO NOT EDIT what you see in these
            // blocks of generated code !
    //}} AFX_MSG_MAP
END_MESSAGE_MAP()
CSqrootApp::CSqrootApp()
{
// The constructor
    // Remove next line for a "quiet" version
    // of MyUser.DLL
    AfxMessageBox("@USER DLL installed");
}
CSqrootApp theApp;
#include <math.h>
extern "C" __declspec(dllexport)
 void MyUser(int* pnNumberOfArgs,
   double* pdArgs, double* dResult)
{
```

```
// This is an @USER routine callable by LINGO. In
// this particular case we simply take the
// square root of the first argument.
   * dResult = sqrt( * pdArgs);
}
```

完成修改后，就可以创建 DLL 了。当 Visual C++ 完成创建过程后，将 SQROOT.DLL 文件复制到 LINGO 的启动目录（LINGO.EXE 所在的文件夹）中，并重新命名为 MYUSER.DLL。这时运行 LINGO 将会看到图 11-9 所示对话框，表明 DLL 已经成功加载。

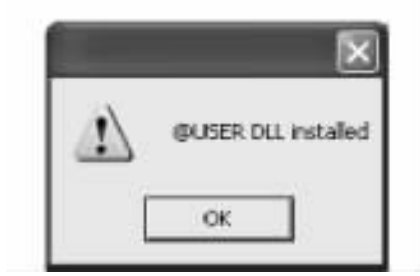



图 11-9　@USER DLL 加载信息框

这时，输入一个小模型计算 9 的方根，求解得到图 11-10 所示的结果。

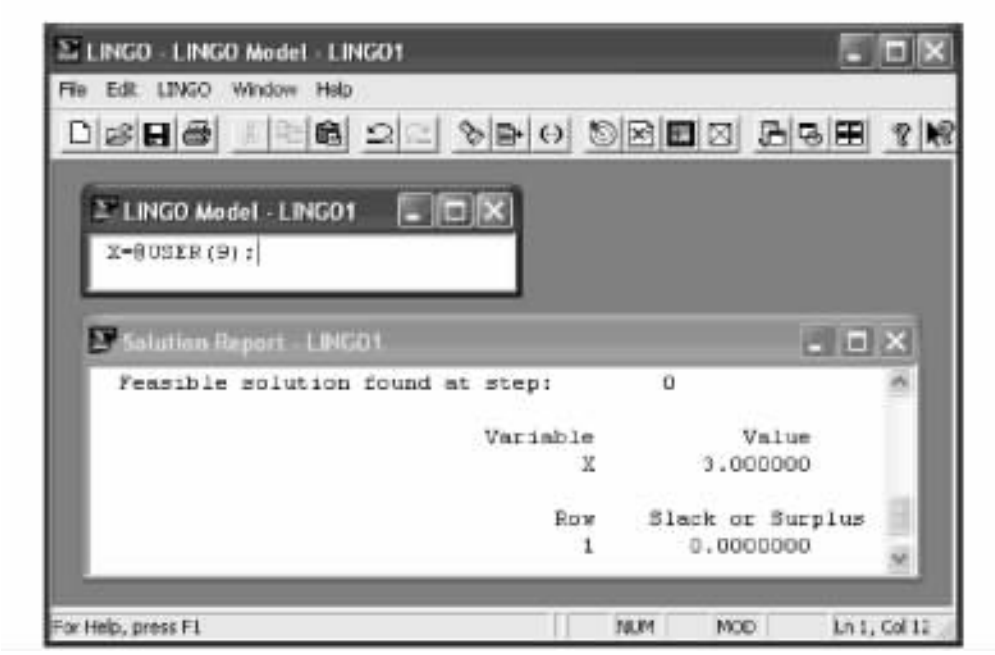



图 11-10　平方根模型求解结果

如果没有 Visual C++，可以直接将 LINGO 提供的 DLL 复制到 LINGO 启动目录中使用上述 @USER 函数。在 LINGO 主目录下的 USER hMSVC 子目录中可以找到 SQROOT.DLL 文件。

# 第 12 章　数 学 模 型

这一章的内容帮助用户了解在 LINGO 中建立模型时，LINGO 算法是如何对模型进行内部处理的。模型的结构会影响求解的速度、方法和解的形式，而 LINGO 的模型具有不同的结构形式。这些也许并不是使用 LINGO 的必要知识，但可以帮助用户有效地使用 LINGO 软件。

## 12.1　LINGO 内部算法的使用

LINGO 具有四种求解方法，用来求解不同类型的模型，它们分别是直接法、线性法、非线性法和分支定界法。

LINGO 的算法与其他模型语言中的算法不同，LINGO 算法和它的模型语言是同一程序的一部分。也就是说，LINGO 算法是与模型语言直接连接起来的。LINGO 与其内部算法的数据是通过内存直接传递的，而不是以中间文件为媒介，这样能够使模型语言组件和算法组件之间的冲突降至最低。

当求解一个模型时，直接算法将首先尽可能多地计算各变量的值。当直接算法发现一个等式约束中只有一个未知变量时，它将会给变量赋予一个满足约束条件的值。当不存在未知变量或不再有任何含有单一未知变量的等式约束时，直接算法的计算就会停止。

直接算法的计算停止后，若所有变量均被计算出结果，LINGO 就会显示结果报告；若还存在未知变量，LINGO 就会检查模型的结构和数学形式，以确定采用何种算法。对于连续的线性模型，LINGO 会使用线性算法对其求解，若模型中含有一个或多个非线性约束，LINGO 会启动非线性算法；若模型含有整型约束，LINGO 则会使用分支定界法。分支定界管理器会根据模型的特点调用线性或非线性算法。

LINGO 中的线性算法使用的是反乘积的修正单纯形法，障碍算法也可用来求解线性模型。LINGO 的非线性算法使用了连续线性规划 (Successive Linear Programming，SLP) 和广义梯度下降 (Generalized Reduced Gradient，GRG) 算法。整数模型使用分支定界法求解。在线性整数模型中，LINGO 会做大量求解前的前处理工作 (如对约束“切削”来限制可行的非整型区域)，这些“切削”将会极大地提高大部分整数模型的求解速度。

## 12.2　约束条件的类型

通过使用直接算法，LINGO 会将模型中所有固定的变量和约束条件替换成已知量，将剩下的变量和约束条件分成线性和非线性类型。LINGO 每次求解模型时打开的求解状态窗口会给出模型中线性、非线性变量和约束条件的数目。若模型中发现任何非线性变量或约束条件，则整个模型是非线性的。这时，将会启动相对比较缓慢的非线性算法替代线性算法。

### 12.2.1 线性约束

如果一个约束的所有项均是一次的，那么这个约束被称为线性约束。也就是说，这个约束中不包含平方、立方和任何高于一次方的项、被变量除的项以及多个变量之间相乘的项，而且变量和约束条件存在一定比例关系。换句话说，每一个变量增加（减少）一个单位，约束条件的值就会增加（减少）一个定值。

线性公式是呈直线关系的。线性公式的基本形式为：

$$Y = mX + b$$

其中，m 和 b 均为定值。

例如，假设买了每磅 1.5 美元的番茄，则计算价钱（C）的表达式或函数是由购买番茄的数量（变量 T）决定的：

$$C = 1.5 \times T$$

可见，价钱函数的图形呈一条直线。

线性表达式可以具有多个变量。例如，假设又多买了每磅 0.75 美元的土豆（P）和每磅 1.25 美元的苹果（A），则价钱函数变成：

$$C = 1.5 \times T + 0.75 \times P + 1.25 \times A$$

这一新的价钱表达式仍然是线性的，可以把它看成是三个简单的线性式的和。因为线性模型比非线性模型运算的速度快而且准确，所以应尽可能地将模型建立为线性表达式。当选中 LINGO|Solve 命令时，LINGO 将分析模型中的内在关系。如果所有的表达式都是线性的，LINGO 将识别并充分利用这一特点。

与其他类型的模型相比，全部用线性关系表达的问题可以更快速地求解。如果 LINGO 能够计算完毕，就会返回最大化问题的目标函数最大值或最小化问题的目标函数最小值。

判断模型中的表达式是否为线性的，方法之一就是查看求解状态窗口中显示的一系列分类统计项目。Nonlinear 目录中的 Variable 框和 Constraints 框中显示了模型中非线性关系的数目。如果这些项均为零，则模型为线性的。

如果 LINGO 模型中存在非线性关系，那么可以研究模型中的约束条件和变量是否可以再形成为线性形式。例如，考虑下面的约束：

$$X / Y = 10$$

显然，这个约束是非线性的，因为 X 被 Y 除。但是该约束可以通过将 Y 移到等式右边而转换为线性约束：

$$X = 10 \times Y$$

### 12.2.2 非线性约束

非线性约束可以定义为不是线性约束的约束条件。非线性约束表达式中的变量为平方或立方项、任何高于一次的项或者变量之间为乘或除的项等。非线性模型较线性模型更难求解。

与线性模型不同,非线性模型即使存在一个解,LINGO 仍可能找不到它。或者,LINGO 可能为一个非线性模型求得一个最优解,但其实仍然存在更优解。这些情况显然不是用户所期望的,要将这些情况发生的可能性降到最低,可以参阅后面的章节。

## 12.3 局部最优和全局最优

当 LINGO 为一个线性最优化模型求得一个解的时候,这个解肯定是最优解,称之为全局最优解。全局最优解是所取得的目标函数值优于模型的其他可行解的可行解,在线性模型中能找到全局最优解要归功于线性模型的特点。

对于非线性最优化问题,情况会有所不同。一个非线性最优化模型可能具有一些局部最优解,所有的非线性算法均收敛于一个局部最优解。也就是说,所求得的解在临近区域内并不存在更优的可行解。虽然在局部最优解的临近区域内找不到更优的解,但是另外的局部最优解可以存在于离当前解较远的区域内,而且这些局部最优点的目标函数值可能更优。因此,当一个非线性模型求解完毕后,所得解仅仅是局部最优解,用户必须意识到其他的局部最优解可能存在,并可能使目标函数值达到更优。

## 12.4 函数的凹凸性

凸性可以提供审视一个解的特性和可靠性的依据。图 12-1 是一个具有单一变量的凸函数。

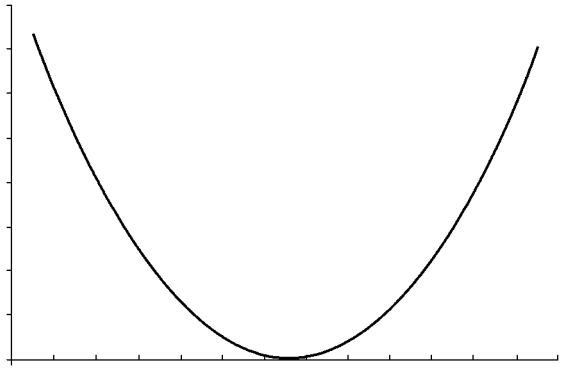


图 12-1 凸函数示意图

凸性的几何学定义为 如果函数中任意两点间的连线均位于该函数之上或与该函数重合,那么这个函数就是凸的。

不含有任何约束条件的凸函数均存在一个单一的最小值区域。可以想像为,雨滴如果沿着曲线流到底部会在全局最低点处形成积水。因此,不管变量的取值范围如何,求解凸函数最小化问题一定会求得全局最优解。如果函数是非凸的,将会存在多于一个的局部最优解。这种情况下,最终解可能为局部最优解而非全局最优解。

确定多变量问题的凸性绝非易事。一个凸函数的数学定义为 :该函数系数矩阵的二次导

数矩阵的行列式值为正值或者它所有的特征值均为正数。如果函数的系数矩阵的二次衍生矩阵的行列式值为负值或它所有的特征值均为负数，则该函数为凹函数。一个函数可以在一个区域内凹，而在另一个区域内凸。例如，X * @COS (3.1416 * X) 函数就是一个凸凹混合函数。图 12-2 中的函数为严格凹函数。

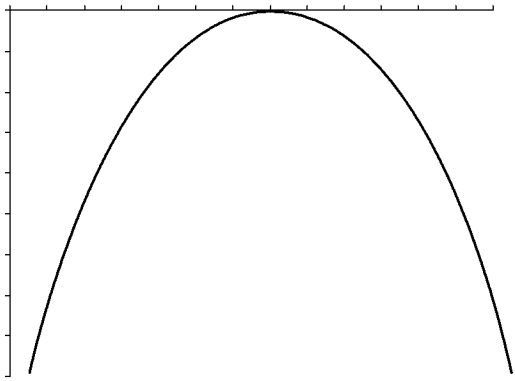


图 12-2　凹函数示意图

## 12.5　平滑和非平滑函数

平滑函数在每一个点都有唯一确定的一阶导数 (斜率或梯度)。从图形上可以看出，含有单一变量的平滑函数可以画成连续的线而没有任何断点或突折。这一章所有例子中的曲线都是平滑的。

非平滑函数包括不可微和非连续函数。如果一个函数的一阶导数存在不确定区域，则称之为不可微函数。不可微函数的图像呈现出突然的弯折状。一个变量的绝对值 (用函数 @ABS (X) 表示) 的图形是一个不可微表达式的例子，如图 12-3 所示。

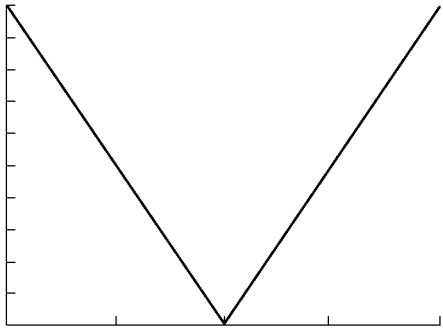


图 12-3　@ABS () 函数图形

在图中，0 点处出现了一个突然的弯折。其他的非平滑函数包括 @MAX、@MIN、@SMAX、@SMIN 以及所有使用线性插值返回非整数参数结果的概率函数。比这种具有突折点的函数更难处理的函数是不连续函数，它们的图形中有很多间断点。LINGO 中的不连续函数有 @SIGN和 @FLOOR。

简单地说，LINGO 会沿着函数的轨迹寻找一个最大或最小点以找到最优解，这一寻找过程用函数导数作为引导。在突折点处导数不存在，这时候 LINGO 处于“失明”状态，不能“看见”突折点周围的点。因此，处理这种含有间断点和突折点的函数较平滑、连续函数困难得多。在可能的情况下，尽量将非平滑函数替换成线性约束问题或线性约束与整型变量的组合问题。

## 12.6 非线性模型的解决方法

从前面的章节中可以看出，非线性模型求解十分困难，因此应该多花一点时间建立利于提高求解效率的模型，以增加求解速度和解的可靠性。这一部分内容将介绍一些建立和求解非线性模型的方法。

### 12.6.1 为变量定界

较好地使用变量界限可以最大限度地提高 LINGO 的求解效率。例如，假设一个变量的取值范围为 1 到 100，但是最优解不大可能在 50 到 75 的范围之外。这种情况下，用 @BND 函数将变量的上限明确地指定为 50，下限为 75，可以大大节省求解的时间。另外，定界还可以使求解避开数学上有问题的区域，如无定义区域。例如，在约束条件 1/X 中，给 X 定一个最小界限，这样有助于使 X 不会靠近 0。

### 12.6.2 为变量指定初值

在模型中，变量的初值会影响 LINGO 求解的“路线”。如果以一个接近最优解的初值开始求解，将大大地减少求解的时间，但是在很多情况下并不知道理想的初值。不过，在可以选定比较合理的初值时，就应该在初始化域使用这些初值，这对于模型的求解很有帮助。

如果存在以下两种对求解结果不确定的情况，可以考虑更换初值重新求解：

1) 可能有比 LINGO 返回值更优的解；

2) 即使 LINGO 返回的信息报告显示“未找到可行解”，但仍可能存在可行解。

### 12.6.3 确定模型的合理数量级范围

为了便于求解，模型中用到的单位的数量级要尽量保持一致。如果模型中最大的数是最小数的 1 000 倍，LINGO 就会在求解模型时遇到困难，也会影响求解的精确度。

例如下面的经济问题 利息率为 8.5% (0.085)，预算的约束条件为 $ 12 850 000。这两个数的数量级相差 $10^9$。而在 LINGO 中，可以接受的最大差别为 $10^4$。在这个例子中，预算可以表示为百万美元的倍数，也就是说可以用 $ 12.85 代替 12 850 000。这样，数量级的差别就控制在 $10^4$ 之内了。

### 12.6.4 简化关系

在实际操作中，应尽量使用线性关系，而非非线性关系。一些非线性表达式完全可以用线性关系表示。最简单的例子就是前面提到的两个变量比值的形式：

$$X/Y < 10$$

这个约束条件是非线性的,因为 X 被 Y 除。为了线性化这一约束,可以在等式两边同乘以 Y,该约束就变为:

$$X < 10 \times Y$$

在可能的情况下要尽量回避使用非平滑关系。具有非平滑约束的模型一般来说求解十分困难。因此应尽量将非平滑关系平滑化,如使用整型变量。

### 12.6.5 减少整型限制

将整型限制的数量降到最低可以大大减少求解时间。在具有较大变量值的时候,可以通过求解没有整型限制的模型后取整找到可以接受的答案,其花费的时间只相当于求解整数模型的一小部分。然而,对一个解取整后,此解可能不再是可行解或最优解。

# 第 13 章　LINGO 在环境系统优化中的应用

通过前面章节的介绍，可以体会到 LINGO 在最优化问题的建模和求解方面具有非常强大的功能，它可以大大提高各个领域的科研工作者在解决系统优化问题时的效率。根据笔者的体会，以往在进行环境系统优化的研究时，除了要建立相应的模型外，还要花费相当大的精力考虑模型的求解问题，而且常常由于模型的形式较为复杂而找不到理想的求解方法。有时为了求解的方便会对模型进行简化，建立相对简单的线性模型，从而不能很好地描述所要解决的问题，也就达不到预期的效果。本章收集了一些环境系统优化领域中的典型问题，尝试用 LINGO 对其进行求解，验证 LINGO 软件的有效性。虽然这些模型不是非常复杂，但是可以为用户提供一些启发，使用户掌握基本方法，以便今后使用 LINGO 解决类似问题。

## 13.1　污染企业的生产安排问题

### 13.1.1　问题描述

某地区有四个化工厂，都生产相同数量的两种化工产品Ⅰ和Ⅱ。这两种产品每 1 000 吨分别获利 20 万元和 30 万元。由于生产过程中排放有害气体，环保部门责令限期治理。为了保护环境和保证各厂不亏损，主管局向各厂进行环保贷款，数额见表 13-1，经测算每生产 1 000 吨产品的环保费用也列于表中。现在的问题是如何安排生产既能保护环境又可使厂家获利最大。

表 13-1　费用表

| 环保费用（$10^4$元/kt） / 产品 | 厂名 | | | |
|---|---|---|---|---|
| | A | B | C | D |
| Ⅰ | 2 | 2 | 4 | 1 |
| Ⅱ | 2 | 1 | 0 | 2 |
| 贷款数额（$10^4$元） | 14 | 10 | 16 | 12 |

根据题意，建立如下数学模型：

目标函数 $\max S = 2x_1 + 3x_2$

s.t.

$2x_1 + 2x_2 \leqslant 14$

$2x_1 + x_2 \leqslant 10$

$4x_1 \leqslant 16$

$$x_1 + 2x_2 \leqslant 12$$

式中：$x_1$，$x_2$分别为Ⅰ、Ⅱ两种产品的千吨数；S 为总利润，单位为 $10^5$元。

## 13.1.2 LINGO 模型

将上述数学模型转化为下列 LINGO 模型：

```
SETS:
  !生产厂家;
  PLANT/P1 P2 P3 P4/:CREDIT,COSTA,COSTB;
  !产品;
  PRODUCTION/A B/:COUNT,GAIN;
ENDSETS
DATA:
  !贷款;
  CREDIT = 14 10 16 12;
  !环保费用;
  COSTA  = 2 2 4 1;
  COSTB  = 2 1 0 2;
  !利润;
  GAIN   = 2 3;
ENDDATA
!目标函数;
[TOTAL_GAIN] MAX = @SUM(PRODUCTION(I):GAIN(I) * COUNT(I));
!贷款约束;
@FOR(PLANT(I):COSTA(I) * COUNT(1) + COSTB(I) * COUNT(2) < = CREDIT(I));
```

## 13.1.3 求解结果

求解上述模型，得到下列结果：

```
Global optimal solution found at iteration:              3
Objective value:                                   190.0000
                     Variable           Value        Reduced Cost
                   CREDIT(P1)        14.00000            0.000000
                   CREDIT(P2)        10.00000            0.000000
                   CREDIT(P3)        16.00000            0.000000
                   CREDIT(P4)        12.00000            0.000000
                    COSTA(P1)        2.000000            0.000000
                    COSTA(P2)        2.000000            0.000000
                    COSTA(P3)        4.000000            0.000000
                    COSTA(P4)        1.000000            0.000000
                    COSTB(P1)        2.000000            0.000000
                    COSTB(P2)        1.000000            0.000000
```

| | | |
|---|---|---|
| COSTB (P3) | 0.000000 | 0.000000 |
| COSTB (P4) | 2.000000 | 0.000000 |
| COUNT (A) | 2.000000 | 0.000000 |
| COUNT (B) | 5.000000 | 0.000000 |
| GAIN (A) | 20.00000 | 0.000000 |
| GAIN (B) | 30.00000 | 0.000000 |
| Row | Slack or Surplus | Dual Price |
| TOTAL_GAIN | 190.0000 | 1.000000 |
| 2 | 0.000000 | 5.000000 |
| 3 | 1.000000 | 0.000000 |
| 4 | 8.000000 | 0.000000 |
| 5 | 0.000000 | 10.00000 |

可见,当Ⅰ、Ⅱ两种产品的产量分别为 2 000 吨和 5 000 吨时,四个化工厂的总利润最高,为 190 万元。由于该模型为线性模型,所以 LINGO 找到了全局最优解。

## 13.2 水处理最优方案问题

### 13.2.1 问题描述

某河段沿岸有三个工厂 A、B、C,均向河中排放污水,形成了水体污染。环保局要求三个工厂共同协商进行污水处理,并规定处理后的污水中 $BOD_5$ 值不能超过 1.6 mg/L (三厂总值)。三个工厂共同研究后,得出三个处理等级方案:方案 1 是作一级处理;方案 2 是作一级与二级处理;方案 3 是作一级、二级和三级处理。各污水处理等级方案所需的费用以及处理后的 $BOD_5$ 值如表 13-2 所示。现在的问题是如何确定处理方案可使处理总费用最少。

表 13-2 各方案的 $BOD_5$ 值与处理费用

| 处理方案 | | $BOD_5$ 值 (mg/L) | | | 费用 (万元) | | |
|---|---|---|---|---|---|---|---|
| 方案号 | 说明 | A厂 | B厂 | C厂 | A厂 | B厂 | C厂 |
| 0 | 无处理 | 1.2 | 0.8 | 1.6 | 0 | 0 | 0 |
| 1 | 一级处理 | 0.6 | 0.6 | 1.0 | 8 | 4 | 10 |
| 2 | 一级和二级处理 | 0.2 | 0.4 | 0.6 | 14 | 8 | 15 |
| 3 | 一级、二级和三级处理 | 0 | 0 | 0 | 17 | 14 | 22 |

### 13.2.2 LINGO 模型

根据题意,建立如下 LINGO 模型:

```
SETS:
  !工厂;
```

```
  FACTORY/1..3/;
  !处理方案;
  METHOD/1..4/;
  !决策方案;
  SCHEME(FACTORY,METHOD) :COST, BOD_OUT, SELECTION;
ENDSETS
DATA:
  !处理费用;
     COST = 0 8 14 17
          0 4 8 14
  0 10 15 22;
  !出水 BOD 值;
    BOD_OUT = 1.2 0.6 0.2 0
              0.8 0.6 0.4 0
                  1.6 1.0 0.6 0;
ENDDATA
MIN = @SUM(SCHEME :COST * SELECTION);
@FOR(SCHEME :@BIN(SELECTION));
@FOR(FACTORY(I) :@SUM(SCHEME(I,J) :SELECTION(I,J)) = 1);
@SUM(SCHEME :BOD_OUT * SELECTION) < = 1.6;
```

## 13.2.3 求解结果

**求解上述模型,得到下列结果:**

```
Global optimal solution found at iteration:           0
Objective value:                                  29.00000
                    Variable           Value        Reduced Cost
                  COST(1, 1)        0.000000            0.000000
                  COST(1, 2)        8.000000            0.000000
                  COST(1, 3)        14.00000            0.000000
                  COST(1, 4)        17.00000            0.000000
                  COST(2, 1)        0.000000            0.000000
                  COST(2, 2)        4.000000            0.000000
                  COST(2, 3)        8.000000            0.000000
                  COST(2, 4)        14.00000            0.000000
                  COST(3, 1)        0.000000            0.000000
                  COST(3, 2)        10.00000            0.000000
                  COST(3, 3)        15.00000            0.000000
                  COST(3, 4)        22.00000            0.000000
              BOD_OUT(1, 1)        1.200000            0.000000
              BOD_OUT(1, 2)       0.6000000            0.000000
              BOD_OUT(1, 3)       0.2000000            0.000000
```

| | | |
|---|---|---|
| BOD_OUT(1,4) | 0.000000 | 0.000000 |
| BOD_OUT(2,1) | 0.8000000 | 0.000000 |
| BOD_OUT(2,2) | 0.6000000 | 0.000000 |
| BOD_OUT(2,3) | 0.4000000 | 0.000000 |
| BOD_OUT(2,4) | 0.000000 | 0.000000 |
| BOD_OUT(3,1) | 1.600000 | 0.000000 |
| BOD_OUT(3,2) | 1.000000 | 0.000000 |
| BOD_OUT(3,3) | 0.6000000 | 0.000000 |
| BOD_OUT(3,4) | 0.000000 | 0.000000 |
| SELECTION(1,1) | 0.000000 | 0.000000 |
| SELECTION(1,2) | 0.000000 | 8.000000 |
| SELECTION(1,3) | 1.000000 | 14.00000 |
| SELECTION(1,4) | 0.000000 | 17.00000 |
| SELECTION(2,1) | 1.000000 | 0.000000 |
| SELECTION(2,2) | 0.000000 | 4.000000 |
| SELECTION(2,3) | 0.000000 | 8.000000 |
| SELECTION(2,4) | 0.000000 | 14.00000 |
| SELECTION(3,1) | 0.000000 | 0.000000 |
| SELECTION(3,2) | 0.000000 | 10.00000 |
| SELECTION(3,3) | 1.000000 | 15.00000 |
| SELECTION(3,4) | 0.000000 | 22.00000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 29.00000 | -1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |

## 13.3 成本—效益问题

### 13.3.1 问题描述

一个城市奉命要削减排入湖泊的城市污水中磷的量。该城市现有三个污水处理厂,排入湖泊的磷的量分别是 1 000 kg/d、500 kg/d 和 2 000 kg/d。现在,要求这三个污水处理厂排放磷的总量削减到 1 000 kg /d 以下。设 $X_t$ 为处理厂 t 增加处理设施后的磷去除百分率,三个污水处理厂的处理费用分别为 $15X_1^2$、$10X_2^2$ 和 $20X_3^2$。求达到磷排放标准的最小污水处理费用。

根据题意,将污水处理厂磷的去除效率 $X_t$ 作为决策变量,则约束条件为:

$$1\,000(1 - X_1/100) + 500(1 - X_2/100) + 2\,000(1 - X_3/100) \leqslant 1\,000$$

简化后为:

$$10X_1 + 5X_2 + 20X_3 \geqslant 2\,500$$

上式表示三个污水处理厂每天必须至少要去除 2 500 kg 的磷，这个去除量可以看作是“资源”而分配到三个处理厂。因此，得到以下的最优化模型：

$$\min Z = 15X_1^2 + 10X_2^2 + 20X_3^2$$

$$\text{s.t.}$$

$$10X_1 + 5X_2 + 20X_3 \geqslant 2\,500$$

$$X_1 \leqslant 100$$

$$X_2 \leqslant 100$$

$$X_3 \leqslant 100$$

## 13.3.2 LINGO 模型

根据题意，建立如下 LINGO 模型：

```
SETS:
  PLANTS/A B C/ :PERCENTAGE, DISCHARGE, COST_COEFFICIENT;
ENDSETS
DATA:
  DISCHARGE = 1000 500 2000;
  COST_COEFFICIENT = 15 10 20;
ENDDATA
MIN = @SUM(PLANTS(I) :COST_COEFFICIENT(I) * PERCENTAGE(I) * PERCENTAGE(I));
@SUM(PLANTS :DISCHARGE * (1 - PERCENTAGE/100)) <= 1000;
@FOR(PLANTS :PERCENTAGE <= 100);
```

可见，使用 LINGO 求解最优化模型时，用户不必对数学模型进行简化，而可以采用模型最原始的形式。这一点对于变量较多、形式较为复杂的模型来说更为重要。

## 13.3.3 求解结果

求解上述模型，得到下列结果：

```
Local optimal solution found at iteration:               48
Objective value:                                  214285.7

                     Variable           Value        Reduced Cost
           PERCENTAGE(A)              57.14286            0.000000
           PERCENTAGE(B)              42.85714        -0.6558638E-06
           PERCENTAGE(C)              85.71429            0.000000
            DISCHARGE(A)              1000.000            0.000000
            DISCHARGE(B)              500.0000            0.000000
            DISCHARGE(C)              2000.000            0.000000
      COST_COEFFICIENT(A)             15.00000            0.000000
      COST_COEFFICIENT(B)             10.00000            0.000000
      COST_COEFFICIENT(C)             20.00000            0.000000

                     Row    Slack or Surplus      Dual Price
```

| | | |
|---|---|---|
| 1 | 214285.7 | -1.000000 |
| 2 | 0.000000 | 171.4286 |
| 3 | 42.85714 | 0.000000 |
| 4 | 57.14286 | 0.000000 |
| 5 | 14.28571 | 0.000000 |

## 13.4 工业废水处理问题

### 13.4.1 问题描述

某金属精炼厂每生产 1 kg 金属产生 3 kg 废物，这些废物随工厂的废水排出，浓度为 2 kg/$m^3$。废水经过部分处理后，排入附近的河流。政府对该厂规定的废物排放标准是 100 000 kg/周。工厂每周最多生产金属 55 000 kg，金属的售价是 1.30 美元/kg，生产成本为 0.90 美元/kg。厂内废水处理设施的处理能力为 70 000 $m^3$/周，处理费用是 0.20 美元/$m^3$。处理效率与污染负荷有关，设 W 代表废水处理量 (10 000 $m^3$/周)。当 W 的变化范围为 0 到 7 时，处理效率等于 1 - 0.06W。求确定最优的达到排放标准的生产和处理方案。

### 13.4.2 LINGO 模型

根据题意，建立如下 LINGO 模型：

```
MAX = P*(1.3-0.9)-W*10000*0.2;
P <= 55000;
W <= 7;
P*3/2 >= W;
P*3 - W*10000*(1-0.06*W)*2 <= 100000;
```

### 13.4.3 求解结果

求解上述模型，得到下列结果：

```
Local optimal solution found at iteration:           100
Objective value:                              14027.78

                    Variable           Value        Reduced Cost
                           P        45486.11            0.000000
                           W        2.083333        -0.7108588E-07

                         Row    Slack or Surplus      Dual Price
                           1        14027.78            1.000000
                           2        9513.891            0.000000
                           3        4.916667            0.000000
                           4        68227.08            0.000000
                           5        0.000000            0.1333333
```

## 13.5 河流污染控制问题

### 13.5.1 问题描述

某城市为改善河流水质状况，要对城市污水汇水区四个排放点的入河有机污染物进行削减。相应的四段城市河道的基本条件如图 13-1 所示。

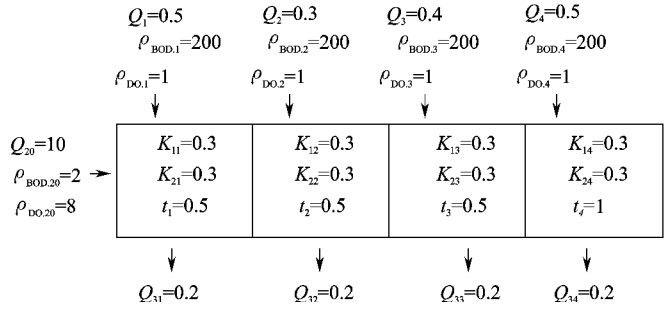



图 13-1 城市四河段排污与水质关系图

图中各量说明如下：

$Q_i$ ——各排污点注入河流的污水流量，mg/s；

$Q_{3i}$ ——由断面 i 引走的流量，mg/s；

$Q_{20}$ ——上游来水水量，mg/s；

$\rho_{BOD.i}$，$\rho_{DO.i}$ ——各节点污水 $BOD_5$、DO 浓度，mg/L；

$\rho_{BOD.20}$，$\rho_{DO.20}$ ——上游来水水质中 $BOD_5$、DO 的浓度，mg/L；

$K_{1i}$，$K_{2i}$ ——各河段 $BOD_5$ 衰减系数、DO 复氧系数，1/d；

$t_i$ ——河水在河段 i 的流经时间，d。

在本题中，i = 1，2，3，4。

河流水质规划的目标预定是：$BOD_5$ 浓度最高为 5 mg/L，DO 浓度最低为 6 mg/L。相应各汇水区内污水处理厂的费用函数为：

$$C_i = 200Q_i^{0.8} + 1\ 000Q_i^{0.8}\eta_i^2 \qquad (i = 1,2,3,4)$$

式中：$\eta_i$ 为河段 i 的污水处理率；其他符号意义同前。

现在的问题是如何选择一最佳城市污水治理方案，既满足河流水质规划目标，又使城市的污水治理总费用最低。

1. 建立费用函数（目标函数）

将 $Q_i$（i = 1，2，3，4）分别代入各个汇水区污水处理治理费用关系式中，得到：

$$C_1 = 115 + 574\eta_1^2$$

$$C_2 = 76.3 + 382\eta_1^2$$

$$C_3 = 96 + 480\eta_1^2$$

$$C_4 = 115 + 574\eta_1^2$$

则该城市污水治理费用为 $C_1 + C_2 + C_3 + C_4$。

2.建立水质约束关系

根据所给河流各段的输入输出信息,利用 S-P 水质方程按下式建立水质关系式:

$$U\rho_{BOD} + m \leqslant m_0$$

$$V\rho_{BOD} + n \geqslant n_0$$

式中:U——$BOD_5$稳定响应矩阵;

V——DO 稳定响应矩阵;

m——$BOD_5$的常数向量;

n——DO 的常数向量;

$m_0$——$BOD_5$ 的约束向量,$m_0 = [5 \quad 5 \quad 5 \quad 5]^T$;

$n_0$——DO 的约束向量,$n_0 = [6 \quad 6 \quad 6 \quad 6]^T$。

经计算得:

$$U = \begin{bmatrix} 0.048\,54 & 0 & 0 & 0 \\ 0.034\,83 & 0.031\,25 & 0 & 0 \\ 0.024\,77 & 0.222\,3 & 0.040\,0 & 0 \\ 0.017\,38 & 0.015\,60 & 0.028\,07 & 0.052\,63 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -0.090\,3 & 0 & 0 & 0 \\ -0.090\,3 & 0 & 0 & 0 \\ -0.011\,18 & -0.005\,759 & 0 & 0 \\ -0.010\,32 & -0.007\,038 & -0.007\,276 & 0 \end{bmatrix}$$

$$m = [1.637\,8 \quad 1.175\,3 \quad 0.0835\,9 \quad 0.586\,7]^T$$

$$n = [7.925\,3 \quad 8.311\,0 \quad 8.533\,5 \quad 8.611\,8]^T$$

并且,$BOD_5$的污水处理率和出口 $BOD_5$的浓度有如下关系:

$$\eta_i = 1 - \rho_{BOD,i}/200$$

3.建立排污口最优规划模型

如果以污水治理总费用最小为目标,满足水质目标要求作为约束条件,并考虑污水处理效率的技术条件,可构造如下 $BOD_5$ 削减模型:

$$\min Z = \sum_{i=1}^{4} C_i$$

$$\text{s.t.}$$

$$U\rho_{BOD} + m \leqslant m_0$$

$$V\rho_{BOD} + n \geqslant n_0$$

$$0 \leqslant \eta_i \leqslant 1$$

$$\rho_{BOD} \geqslant 0$$

## 13.5.2 LINGO 模型

根据题意,建立如下 LINGO 模型:

```
SETS:
  RIVERS/A,B,C,D/:OUTBOD,RIVERDO,RIVERBOD,M,N,RATE,COST,
BODMAX,DOMIN,A1,A2;
  RIVERPOINTS(RIVERS,RIVERS):U,V;
ENDSETS
DATA:
  M=1.6378,1.1753,0.08359,0.5867;
  N=7.9253,8.3110,8.5335,8.6118;
  U=0.04854 0 0 0
    0.03483 0.03125 0 0
    0.02477 0.02223 0.0400 0
    0.01738 0.01560 0.02807 0.05263;
  V=0 0 0 0
    -0.0903 0 0 0
    -0.01118 -0.005759 0 0
    -0.01032 -1.007038 -0.007276 0;
  A1=115 76.3 96 115;
  A2=574 382 480 574;
  BODMAX=5 5 5 5;
  DOMIN=6 6 6 6;
ENDDATA
MIN=@SUM(RIVERS:COST);
@FOR(RIVERS(I):COST(I)=A1(I)+A2(I)*RATE(I)*RATE(I));
@FOR(RIVERS(I):RATE(I)=1-OUTBOD(I)/200);
@FOR(RIVERS(I):(@SUM(RIVERS(J):U(I,J)*OUTBOD(J))+M(I))<=BODMAX(I));
@FOR(RIVERS(I):(@SUM(RIVERS(J):V(I,J)*OUTBOD(J))+N(I))>=DOMIN(I));
@FOR(RIVERS(I):RIVERBOD(I)=@SUM(RIVERS(J):U(I,J)*OUTBOD(J))+M(I));
@FOR(RIVERS(I):RIVERDO(I)=@SUM(RIVERS(J):V(I,J)*OUTBOD(J))+N(I));
@FOR(RIVERS(I):RATE(I)<=1);
@FOR(RIVERS(I):RATE(I)>=0);
```

## 13.5.3 求解结果

**求解上述模型,得到下列结果:**

```
Local optimal solution found at iteration:              25
Objective value:                                  1672.786

                     Variable           Value        Reduced Cost
                    OUTBOD(A)        25.59247            0.000000
                    OUTBOD(B)        93.86606            0.000000
                    OUTBOD(C)        54.89605            0.000000
                    OUTBOD(D)        18.30249            0.000000
                   RIVERDO(A)        7.925300            0.000000
```

| | | |
|---|---|---|
| RIVERDO (B) | 6.000000 | 0.000000 |
| RIVERDO (C) | 7.706802 | 0.000000 |
| RIVERDO (D) | 7.287633 | 0.000000 |
| RIVERBOD (A) | 2.880058 | 0.000000 |
| RIVERBOD (B) | 5.000000 | 0.000000 |
| RIVERBOD (C) | 5.000000 | 0.000000 |
| RIVERBOD (D) | 5.000000 | 0.000000 |
| M (A) | 1.637800 | 0.000000 |
| M (B) | 1.175300 | 0.000000 |
| M (C) | 0.8359000E-01 | 0.000000 |
| M (D) | 0.5867000 | 0.000000 |
| N (A) | 7.925300 | 0.000000 |
| N (B) | 8.311000 | 0.000000 |
| N (C) | 8.533500 | 0.000000 |
| N (D) | 8.611800 | 0.000000 |
| RATE (A) | 0.8720377 | 0.000000 |
| RATE (B) | 0.5306697 | 0.000000 |
| RATE (C) | 0.7255197 | 0.000000 |
| RATE (D) | 0.9084875 | 0.000000 |
| COST (A) | 551.4981 | 0.000000 |
| COST (B) | 183.8752 | 0.000000 |
| COST (C) | 348.6619 | 0.000000 |
| COST (D) | 588.7507 | 0.000000 |
| BODMAX (A) | 5.000000 | 0.000000 |
| BODMAX (B) | 5.000000 | 0.000000 |
| BODMAX (C) | 5.000000 | 0.000000 |
| BODMAX (D) | 5.000000 | 0.000000 |
| DOMIN (A) | 6.000000 | 0.000000 |
| DOMIN (B) | 6.000000 | 0.000000 |
| DOMIN (C) | 6.000000 | 0.000000 |
| DOMIN (D) | 6.000000 | 0.000000 |
| A1 (A) | 115.0000 | 0.000000 |
| A1 (B) | 76.30000 | 0.000000 |
| A1 (C) | 96.00000 | 0.000000 |
| A1 (D) | 115.0000 | 0.000000 |
| A2 (A) | 574.0000 | 0.000000 |
| A2 (B) | 382.0000 | 0.000000 |
| A2 (C) | 480.0000 | 0.000000 |
| A2 (D) | 574.0000 | 0.000000 |
| U (A, A) | 0.4854000E-01 | 0.000000 |
| ⋮ | | |
| V (D, D) | 0.000000 | 0.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 1672.786 | -1.000000 |
| 2 | 0.000000 | -1.000000 |
| 3 | 0.000000 | -1.000000 |
| 4 | 0.000000 | -1.000000 |
| 5 | 0.000000 | -1.000000 |
| 6 | 0.000000 | -1001.099 |
| 7 | 0.000000 | -405.4318 |
| 8 | 0.000000 | -696.4991 |
| 9 | 0.000000 | -1042.944 |
| 10 | 2.119942 | 0.000000 |
| 11 | 0.000000 | 2.936071 |
| 12 | 0.000000 | 17.53114 |
| 13 | 0.000000 | 99.08265 |
| 14 | 1.925300 | 0.000000 |
| 15 | 0.000000 | -30.42006 |
| 16 | 1.706802 | 0.000000 |
| 17 | 1.287633 | 0.000000 |
| 18 | 0.000000 | 0.000000 |
| 19 | 0.000000 | 0.000000 |
| 20 | 0.000000 | 0.000000 |
| 21 | 0.000000 | 0.000000 |
| 22 | 0.000000 | 0.000000 |
| 23 | 0.000000 | 0.000000 |
| 24 | 0.000000 | 0.000000 |
| 25 | 0.000000 | 0.000000 |
| 26 | 0.1279623 | 0.000000 |
| 27 | 0.4693303 | 0.000000 |
| 28 | 0.2744803 | 0.000000 |
| 29 | 0.9151247E-01 | 0.000000 |
| 30 | 0.8720377 | 0.000000 |
| 31 | 0.5306697 | 0.000000 |
| 32 | 0.7255197 | 0.000000 |
| 33 | 0.9084875 | 0.000000 |

## 13.6 水资源开发利用问题

### 13.6.1 问题描述

**图 13-2 为一个简单的流域系统。该系统中有两个水库和一个水电厂，从水库出来的水不仅用于水力发电，还用于灌溉。各水文区段上的来水量有丰、枯季节之分（上面一行数字表示**

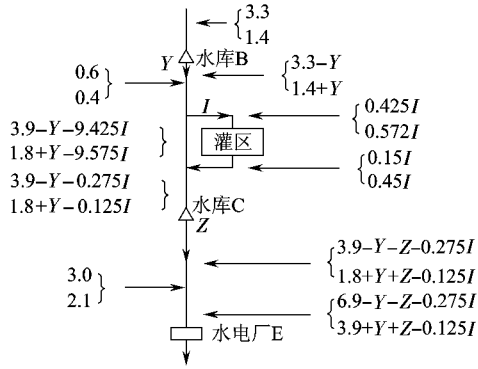



**图 13-2　多目标水资源开发工程示意图**

丰水季节来水量，下面一行数字表示枯水季节来水量，单位均为 $10^9\ m^3$），并且已知：

$$B_1(E) = 320E + 2E^2$$

$$B_2(I) = 45.4I_1 + 15.3I_2 + 1\,045\lg(1 + 0.2I)$$

$$K_1(Y) = \frac{43Y}{(1 + 0.2Y)}$$

$$K_2(Z) = \frac{47Z}{(1 + 0.3Z)}$$

$$K_3(E) = 20.6E - E^2$$

$$K_4(I) = 44I_1 + 64I_2 + 4.5I_1^+ + 0.5I_2^+$$

式中：Y——B 水库的有效库容，$10^9 m^3$；

Z——C 水库的有效库容，$10^9 m^3$；

I——年灌溉用水量，$10^9 m^3$；

$B_1(E)$——电力现值，$10^6$ 美元；

$B_2(I)$——灌溉用水现值，$10^6$ 美元；

$K_1(Y)$——B 水库有效库容为 $Y(10^9 m^3)$ 时的建库投资费用，$10^6$ 美元；

$K_2(Z)$——C 水库有效库容为 $Z(10^9 m^3)$ 时的建库投资费用，$10^6$ 美元；

$K_3(E)$——建造年发电量为 $E \times 10^9$ 度的水电厂的投资费用，$10^6$ 美元；

$K_4(I)$——建造年灌溉用水规模为 $I \times 10^9 m^3$ 的灌溉系统的投资费用，$10^6$ 美元。

关于 $K_4(I)$，由于灌溉用水量超过 $3 \times 10^9\ m^3$，故需要抽水灌溉。因此，投资费用函数将分为两部分 表示自流灌溉部分的 $I_1$ 和表示抽水灌溉部分的 $I_2$。式中 $I_1 + I_2 = I$，$I_1 \leqslant 3$。$I^+$ 是一个函数，当 $I > 0$ 时，$I^+$ 值为 1，否则等于零。

本题将结构尺寸和不同配水方案的目标输出作为变量，在满足给定的水文和技术方面的约束条件下，求能使经济目标函数为最大值的最优结构尺寸和最优目标输出。

根据题意，建立如下数学模型：

目标函数：$\max \pi = B_1(E) + B_2(I) - K_1(Y) - K_2(Z) - K_3(E) - K_4(I)$

s.t.

$3.3 - Y \geqslant 0$

$$3.9 - Y - 0.425I \geqslant 0$$

$$1.8 + Y - 0.575 \geqslant 0$$

$$3.9 - Y - Z - 0.275 \geqslant 0$$

$$Y + Z + 0.275I + 3.47E \leqslant 6.9$$

$$- Y - Z + 0.25I + 3.47E \leqslant 3.0$$

**式中：$\pi$ 为系统总收益（$10^6$美元）；其他符号意义同前。**

**并且目标函数可以表示为：**

$$\max \pi = 229.4E + E^2 + 1.4I_1 - 48.7I_2 + 1\,045\lg(1 + 0.2I) - 4.5I_1^+ - 0.5I_2^+ - \frac{43Y}{(1 + 0.2Y)} - \frac{47Z}{(1 + 0.3Z)}$$

## 13.6.2 LINGO 模型

**将上述数学模型转化为下列 LINGO 模型：**

```
max = B1 + B2 - K1 - K2 - K3 - K4;
B1 = 320 * E + 2 * E * E;
B2 = 45.4 * I1 + 15.3 * I2 + 1045 * @LOG(1 + 0.2 * I);
K1 = 43 * Y/(1 + 0.2 * I);
K2 = 47 * Z/(1 + 0.3 * Z);
K3 = 20.6 * E-E * E;
K4 = 44 * I1 + 64 * I2 + 4.5 * @IF(I1 # GT # 0,1,0) + 0.5 * @IF(I2 # GT # 0,1,0);
I1 + I2 = I;
3.3 - Y > = 0;
3.9 - Y - 0.425 * I > = 0;
1.8 + Y - 0.575 * I > = 0;
3.9 - Y-Z - 0.275 * I > = 0;
Y + Z + 0.275 * I + 3.47 * E < = 6.9;
- Y-Z + 0.25 * I + 3.47 * E < = 3.9;
I1 < = 3;
```

**该模型变量众多且不宜采用集合形式表现，所以整个模型显得较为易读。对于非线性模型，LINGO 无需作特别处理。在本例中，要特别注意 @IF() 函数的用法，它使得模型特别直观和简单。**

## 13.6.3 求解结果

**求解上述模型，得到下列结果：**

```
Linearization components added:
  Constraints:          30
  Variables:            20
  Integers:             12
Local optimal solution found at iteration:            837
```

Objective value: 963.0488

| Variable | Value | Reduced Cost |
|---|---|---|
| B1 | 364.5683 | 0.000000 |
| B2 | 957.6204 | 0.000000 |
| K1 | 28.95391 | 0.000000 |
| K2 | 0.000000 | 0.000000 |
| K3 | 22.02452 | 0.000000 |
| K4 | 308.1615 | 0.000000 |
| E | 1.131277 | 0.000000 |
| I1 | 2.816394 | 0.000000 |
| I2 | 2.800627 | 0.000000 |
| I | 5.617021 | 0.000000 |
| Y | 1.429787 | 0.000000 |
| Z | 0.000000 | 77.03798 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 963.0488 | 1.000000 |
| 2 | 0.000000 | 1.000000 |
| 3 | 0.000000 | 1.000000 |
| 4 | 0.000000 | -1.000000 |
| 5 | 0.000000 | -1.000000 |
| 6 | 0.000000 | -1.000000 |
| 7 | 0.000000 | -1.000000 |
| 8 | 0.000000 | -48.70000 |
| 9 | 1.870213 | 0.000000 |
| 10 | 0.8297872E-01 | 0.000000 |
| 11 | 0.000000 | -50.28848 |
| 12 | 0.9255319 | 0.000000 |
| 13 | 0.000000 | 59.13825 |
| 14 | 0.000000 | 29.10027 |
| 15 | 0.1836060 | 50.10000 |

**可见，LINGO 在求解该模型时采用了线性化处理方法，最终找到的解为局部最优解。由于模型是非线性并使用了 @IF() 函数，所以使得求解的时间相对较长。结果显示，不修建水库 C 会使整个系统的收益最大。**

## 13.7 水资源分配问题

### 13.7.1 问题描述

**设水库可分配水资源量为 7 个单位，供给 3 个用户，各用户在分配水量 $q_k$ 下的效益函数为 $g_i(q_k)$，i = 1，2，3，如表 13-3 所示。求水资源量的最优分配方案。**

表 13-3　效益函数表

| $g(q_k)$ \ $q_k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $g_1(q_k)$ | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 100 |
| $g_2(q_k)$ | 0 | 5 | 15 | 40 | 60 | 70 | 73 | 75 |
| $g_3(q_k)$ | 0 | 4 | 26 | 40 | 45 | 50 | 51 | 53 |

**此类题目过去多采用动态规划法逐步计算出最优解,而使用 LINGD 求解相对简便。**

## 13.7.2　LINGO 模型

**根据题意,建立下列 LINGO 模型:**

```
SETS:
  !用户;
  USER/1..3/;
  !水量;
  WATER_AMOUNT/1..8/;
  !分配方案;
  ARCS(USER,WATER_AMOUNT):BENEFIT,SELECTION,STATUS;
ENDSETS
DATA:
!效益;
BENEFIT = 0 5 15 40 80 90 95 100
          0 5 15 40 60 70 73 75
          0 4 26 40 45 50 51 53;
!待定分配量;
STATUS = 0 1 2 3 4 5 6 7
         0 1 2 3 4 5 6 7
         0 1 2 3 4 5 6 7;
   ENDDATA
MAX = @SUM(ARCS(I,J):BENEFIT(I,J) * SELECTION(I,J));
@FOR(ARCS:@BIN(SELECTION));
@FOR(USER(I):@SUM(ARCS(I,K):SELECTION(I,K)) = 1);
@SUM(ARCS(I,J):STATUS(I,J) * SELECTION(I,J)) = 7;
```

**在建立此模型时,使用了一个小技巧,就是为 ARCS 集合添加了 SELECTION 和 STATUS 属性。通过决策变量 SELECTION 和已知量 STATUS 的联合使用,不仅轻松表达了总水量的约束,而且为离散取值最优化问题的求解提供了一个有效的思路。**

## 13.7.3 求解结果

**求解上述模型,得到下列结果：**

Global optimal solution found at iteration： 0

Objective value： 120.0000

| Variable | Value | Reduced Cost |
|---|---|---|
| BENEFIT(1, 1) | 0.000000 | 0.000000 |
| ⋮ | | |
| BIN(1, 1) | 0.000000 | 0.000000 |
| BIN(1, 2) | 0.000000 | -5.000000 |
| BIN(1, 3) | 0.000000 | -15.00000 |
| BIN(1, 4) | 0.000000 | -40.00000 |
| BIN(1, 5) | 1.000000 | -80.00000 |
| BIN(1, 6) | 0.000000 | -90.00000 |
| BIN(1, 7) | 0.000000 | -95.00000 |
| BIN(1, 8) | 0.000000 | -100.0000 |
| BIN(2, 1) | 0.000000 | 0.000000 |
| BIN(2, 2) | 0.000000 | -5.000000 |
| BIN(2, 3) | 0.000000 | -15.00000 |
| BIN(2, 4) | 1.000000 | -40.00000 |
| BIN(2, 5) | 0.000000 | -60.00000 |
| BIN(2, 6) | 0.000000 | -70.00000 |
| BIN(2, 7) | 0.000000 | -73.00000 |
| BIN(2, 8) | 0.000000 | -75.00000 |
| BIN(3, 1) | 1.000000 | 0.000000 |
| BIN(3, 2) | 0.000000 | -4.000000 |
| BIN(3, 3) | 0.000000 | -26.00000 |
| BIN(3, 4) | 0.000000 | -40.00000 |
| BIN(3, 5) | 0.000000 | -45.00000 |
| BIN(3, 6) | 0.000000 | -50.00000 |
| BIN(3, 7) | 0.000000 | -51.00000 |
| BIN(3, 8) | 0.000000 | -53.00000 |
| STATUS(1, 1) | 0.000000 | 0.000000 |
| STATUS(1, 2) | 1.000000 | 0.000000 |
| STATUS(1, 3) | 2.000000 | 0.000000 |
| STATUS(1, 4) | 3.000000 | 0.000000 |
| STATUS(1, 5) | 4.000000 | 0.000000 |
| STATUS(1, 6) | 5.000000 | 0.000000 |
| STATUS(1, 7) | 6.000000 | 0.000000 |
| STATUS(1, 8) | 7.000000 | 0.000000 |

⋮

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 120.0000 | 1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |

可见，将 7 个单位的水资源分别以 4、3、0 的比例分配给 3 个用户可得到最大收益。

## 13.8 水库群最优调度问题

### 13.8.1 问题描述

设水库群系统的组成如图 13-3 所示，要求在 12 个运行期（N = 12）上进行优化，使其总收益最大。已知在任何运行期内进入水库 1 和 2 的流量分别为 input1 和 input2。四个水库在每个运行期的出流为 output（I，J）（I = 0，1，…，11；J = 1，2，3，4），通常用来发电，仅 output（I，4）经发电后，再引水到灌区灌溉。4 个水库的蓄水量分别为 storage（I，J）。

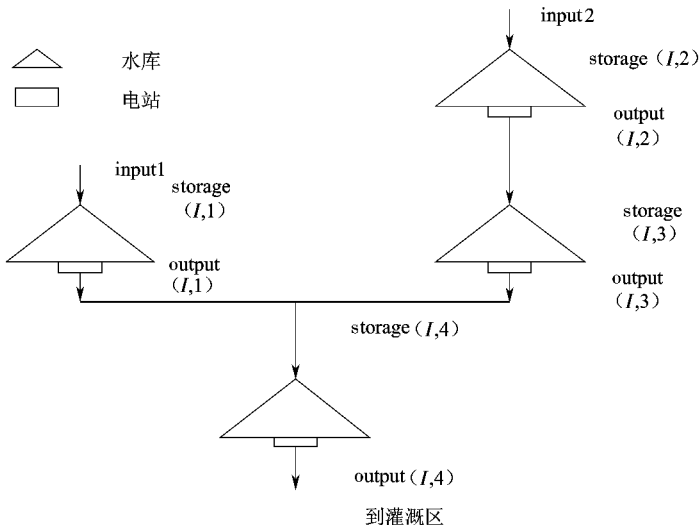



图 13-3 水库群系统组成图

根据已知数据，可知蓄水量的约束条件为：

$$0 \leqslant \text{storage}(I,1) \leqslant 10$$

$$0 \leqslant \text{storage}(I,2) \leqslant 10$$

$$0 \leqslant \text{storage}(I,3) \leqslant 10$$

$$0 \leqslant \text{storage}(I,4) \leqslant 15$$

其中，I = 1，2，…，12。

任何运行期，决策变量的约束条件为：

$0 \leqslant \text{output}(I,1) \leqslant 3$

$0 \leqslant \text{output}(I,2) \leqslant 4$

$0 \leqslant \text{output}(I,3) \leqslant 4$

$0 \leqslant \text{output}(I,4) \leqslant 7$

**其中，**$I = 0,1,\cdots,11$。

**反应任一级** I **的每个分量动态特性的系统方程为：**

$\text{storge}(I,1) = \text{storage}(I-1,1) + \text{input1} - \text{output}(I-1,1)$

$\text{storge}(I,2) = \text{storage}(I-1,2) + \text{input2} - \text{output}(I-1,2)$

$\text{storge}(I,3) = \text{storage}(I-1,3) + \text{input}(I-1,2) - \text{output}(I-1,3)$

$\text{storge}(I,4) = \text{storage}(I-1,4) + \text{input}(I-1,3) + \text{output}(I-1,1) - \text{output}(I-1,4)$

**其中，**$I = 1,2,\cdots,12$。

**所有时段的入流均为：**

$\text{input1} = 2, \text{input2} = 3$；

**上述所有变量和约束均为体积单位。**

**系统的效益是指四个水电站所产生的电能效益与** $\text{output}(I,4)$ **的灌溉效益之和，即：**

$$F = \sum_{I=0}^{11}\sum_{J=1}^{4}(\text{benefit}(I,J) \times \text{output}(I,J)) + \sum_{I=0}^{11}(\text{benefit}(I,5) \times \text{output}(I,4)) + \sum_{J=1}^{4}\text{reservoir}(J)[\text{storage}(I,J), \text{expectation}(I,J)]$$

**式中：**$F$ **——12 个时段的系统总效益；**

$\text{benefit}(I,J)$ **——** $I$ **至** $I-1$ **级时段内的第** $J$ **项用途** $(J = 1,2,\cdots,5)$ **的单位输出所得的效益，在这个问题中共有五项用途，即四个水力发电站和一个灌区，其单位效益函数** $\text{benefit}(I,J)$ **值列于表 13-4 中；**

**表 13-4　单位效益函数**

| I | benefit (I,1) | benefit (I,2) | benefit (I,3) | benefit (I,4) | benefit (I,5) |
|---|---|---|---|---|---|
| 0 | 1.1 | 1.4 | 1 | 1 | 1.6 |
| 1 | 1 | 1.1 | 1 | 1.2 | 1.7 |
| 2 | 1 | 1 | 1.2 | 1.8 | 1.8 |
| 3 | 1.2 | 1 | 1.8 | 2.5 | 1.9 |
| 4 | 1.8 | 1.2 | 2.5 | 2.2 | 2 |
| 5 | 2.5 | 1.8 | 2.2 | 2 | 2 |
| 6 | 2.2 | 2.5 | 2 | 1.8 | 2 |
| 7 | 2 | 2.2 | 1.8 | 2.2 | 1.9 |
| 8 | 1.8 | 2 | 2.2 | 1.8 | 1.8 |
| 9 | 2.2 | 1.8 | 1.8 | 1.4 | 1.7 |
| 10 | 1.8 | 2.2 | 1.4 | 1.1 | 1.6 |
| 11 | 1.4 | 1.8 | 1.1 | 1 | 1.5 |
| 12 | 0 | 0 | 0 | 0 | 0 |

reservoir(J)[storage(I,J), expectation(I,J)]——惩罚函数,是当第 I 级第 J 个分量的最后状态是 storage(I,J)而不是所希望的状态 expectation(I,J)时(J = 1,2,3,4)对系统估计的一个惩罚函数量。

惩罚函数采用下列形式:

$$\text{reservoir}(J)[\text{storage}(I,J), \text{expectation}(I,J)] = -40[\text{storage}(I,J) - \text{expectation}(I,J)]^2$$

$$\text{storage}(I,J) \leqslant \text{expectation}(I,J)$$

$$\text{reservoir}(J)[\text{storage}(I,J), \text{expectation}(I,J)] = 0 \quad \text{storage}(I,J) > \text{expectation}(I,J)$$

式中:expectation(1,J),expectation(12,J)为设定的初级和末级状态向量,其数值分别为:

$$\text{expectation}(1,J) = [5 \quad 5 \quad 5 \quad 5]^T$$

$$\text{expectation}(12,J) = [5 \quad 5 \quad 5 \quad 5]^T$$

## 13.8.2 LINGO 模型

根据题意,建立如下 LINGO 模型:

```
SETS:
  RESERVOIR/P1 P2 P3 P4/:EXPECTATION, MINSTORAGE, MAXSTORAGE,
                         MINOUTPUT, MAXOUTPUT;
  TIME/0 1 2 3 4 5 6 7 8 9 10 11 12/;
  PURPOSE/B1 B2 B3 B4 B5/;
  LINKS1(TIME,RESERVOIR):STORAGE,OUTPUT;
  LINKS2(TIME,PURPOSE):BENEFIT;
ENDSETS
DATA:
  !某级状态期望值,水库存水量最小值,水库存水量最大值,水库流出水量最小值,水库流出水量最大值;
  EXPECTATION,MINSTORAGE,MAXSTORAGE,MINOUTPUT,MAXOUTPUT =
@OLE('D:\LINGO_DATA\数据10.XLS');
  !单位效益;
  BENEFIT = @OLE('D:\LINGO_DATA\数据10.XLS');
  !其他固定参数;
  INPUT1 = 2;INPUT2 = 3;!水库1和水库2的输入流量;
  !输出水库流出水量(T/A);
  @OLE('D:\LINGO_DATA\数据10.XLS') = OUTPUT;
  @OLE('D:\LINGO_DATA\数据10.XLS') = STORAGE;
ENDDATA
!目标函数;
[TOTAL_GAIN] MAX = @SUM(LINKS1(I,J)|I #LE# 12:OUTPUT(I,J) * BENEFIT(I,J))
    + @SUM(TIME(I)|I #LE# 12:OUTPUT(I,4) * BENEFIT(I,5))
    + @SUM(RESERVOIR(J):@IF(STORAGE(12,J) #LE# EXPECTATION(J),
```

```
        - 40 * (STORAGE(12,J) - EXPECTATION(J))^2,0))
        + @SUM(RESERVOIR(J):@IF(STORAGE(1,J) #LE# EXPECTATION(J),
         - 40 * (STORAGE(1,J) - EXPECTATION(J))^2,0));
```

**!需求约束；**

```
@FOR(TIME(I)|I #GT# 1:
  STORAGE(I,1) = STORAGE(I - 1,1) + INPUT1 - OUTPUT(I - 1,1);
  STORAGE(I,2) = STORAGE(I - 1,2) + INPUT2 - OUTPUT(I - 1,2);
  STORAGE(I,3) = STORAGE(I - 1,3) + OUTPUT(I - 1,2) - OUTPUT(I - 1,3);
STORAGE(I,4) = STORAGE(I - 1,4) + OUTPUT(I - 1,3) + OUTPUT(I - 1,1) - OUTPUT(I - 1,4));
```

**!边界约束；**

```
@FOR(LINKS1(I,J):STORAGE(I,J) > = MINSTORAGE(J);STORAGE(I,J) < = MAXSTORAGE(J);
OUTPUT(I,J) > = MINOUTPUT(J);OUTPUT(I,J) < = MAXOUTPUT(J));
```

**本例中的数据存放于 Excel 文件中，如图 13-4、13-5 所示。**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 已知量 | 水库 | | | | |
| 2 | | P1 | P2 | P3 | P4 | |
| 3 | expectation | 5 | 5 | 5 | 5 | |
| 4 | MINstorage | 0 | 0 | 0 | 0 | |
| 5 | MAXstorage | 10 | 10 | 10 | 15 | |
| 6 | MINoutput | 0 | 0 | 0 | 0 | |
| 7 | MAXoutput | 3 | 4 | 4 | 7 | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | 单位效益(benefit) | | | | |
| 11 | 运行期 | b1 | b2 | b3 | b4 | b5 |
| 12 | 0 | 1.1 | 1.4 | 1 | 1 | 1.6 |
| 13 | 1 | 1 | 1.1 | 1 | 1.2 | 1.7 |
| 14 | 2 | 1 | 1 | 1.2 | 1.8 | 1.8 |
| 15 | 3 | 1.2 | 1 | 1.8 | 2.5 | 1.9 |
| 16 | 4 | 1.8 | 1.2 | 2.5 | 2.2 | 2 |
| 17 | 5 | 2.5 | 1.8 | 2.2 | 2 | 2 |
| 18 | 6 | 2.2 | 2.5 | 2 | 1.8 | 2 |
| 19 | 7 | 2 | 2.2 | 1.8 | 2.2 | 1.9 |
| 20 | 8 | 1.8 | 2 | 2.2 | 1.8 | 1.8 |
| 21 | 9 | 2.2 | 1.8 | 1.8 | 1.4 | 1.7 |
| 22 | 10 | 1.8 | 2.2 | 1.4 | 1.1 | 1.6 |
| 23 | 11 | 1.4 | 1.8 | 1.1 | 1 | 1.5 |
| 24 | 12 | 0 | 0 | 0 | 0 | 0 |

**图 13-4　已知数据表**

## 13.8.3　求解结果

**由于结果太长，只列出部分结果如下：**

```
Local optimal solution found at iteration:              126
Objective value:                                  494.6804
  Export Summary Report
```

| 运行期 | 水库流出水量(output) | | | | 运行期 | 水库存储水量(storage) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | | P1 | P2 | P3 | P4 |
| 0 | | | | | 0 | | | | |
| 1 | | | | | 1 | | | | |
| 2 | | | | | 2 | | | | |
| 3 | | | | | 3 | | | | |
| 4 | | | | | 4 | | | | |
| 5 | | | | | 5 | | | | |
| 6 | | | | | 6 | | | | |
| 7 | | | | | 7 | | | | |
| 8 | | | | | 8 | | | | |
| 9 | | | | | 9 | | | | |
| 10 | | | | | 10 | | | | |
| 11 | | | | | 11 | | | | |
| 12 | | | | | 12 | | | | |

**图 13-5　决策变量表**

```
- - - - - - - - - - - - - - - - - - - - - -
Transfer Method:       OLE BASED
Spreadsheet:           D:\LINGO_DATA\数据10.XLS
Ranges Specified:              1
    OUTPUT
Ranges Found:                  1
Range Size Mismatches:         0
Values Transferred:           52
Export Summary Report
- - - - - - - - - - - - - - - - - - - - - -
Transfer Method:       OLE BASED
Spreadsheet:           D:\LINGO_DATA\数据10.XLS
Ranges Specified:              1
    STORAGE
Ranges Found:                  1
Range Size Mismatches:         0
Values Transferred:           52
                Variable           Value        Reduced Cost
                                   ⋮
          STORAGE(0, P1)        10.00000            0.000000
          STORAGE(0, P2)        10.00000            0.000000
          STORAGE(0, P3)        10.00000            0.000000
          STORAGE(0, P4)        15.00000            0.000000
          STORAGE(1, P1)        9.000000            0.000000
          STORAGE(1, P2)        9.000000            0.000000
          STORAGE(1, P3)        10.00000            0.000000
          STORAGE(1, P4)        15.00000            0.000000
          STORAGE(2, P1)        8.000000            0.000000
```

| | | |
|---|---|---|
| STORAGE(2, P2) | 8.949998 | 0.000000 |
| STORAGE(2, P3) | 10.00000 | 0.000000 |
| STORAGE(2, P4) | 14.05000 | 0.000000 |
| STORAGE(3, P1) | 10.00000 | 0.000000 |
| STORAGE(3, P2) | 9.379697 | 0.000000 |
| STORAGE(3, P3) | 8.570300 | 0.000000 |
| STORAGE(3, P4) | 11.05000 | 0.000000 |
| STORAGE(4, P1) | 10.00000 | 0.000000 |
| STORAGE(4, P2) | 10.00000 | 0.000000 |
| STORAGE(4, P3) | 6.949998 | 0.000000 |
| STORAGE(4, P4) | 10.05000 | 0.000000 |
| STORAGE(5, P1) | 9.000000 | 0.000000 |
| STORAGE(5, P2) | 10.00000 | 0.000000 |
| STORAGE(5, P3) | 5.949998 | 0.000000 |
| STORAGE(5, P4) | 10.05000 | 0.000000 |
| STORAGE(6, P1) | 8.000000 | 0.000000 |
| STORAGE(6, P2) | 9.000000 | 0.000000 |
| STORAGE(6, P3) | 5.949998 | 0.000000 |
| STORAGE(6, P4) | 10.05000 | 0.000000 |
| STORAGE(7, P1) | 7.000000 | 0.000000 |
| STORAGE(7, P2) | 8.000000 | 0.000000 |
| STORAGE(7, P3) | 5.949998 | 0.000000 |
| STORAGE(7, P4) | 10.05000 | 0.000000 |
| STORAGE(8, P1) | 6.000000 | 0.000000 |
| STORAGE(8, P2) | 7.000000 | 0.000000 |
| STORAGE(8, P3) | 5.949998 | 0.000000 |
| STORAGE(8, P4) | 10.05000 | 0.000000 |
| STORAGE(9, P1) | 6.977499 | 0.000000 |
| STORAGE(9, P2) | 6.000000 | 0.000000 |
| STORAGE(9, P3) | 5.949998 | 0.000000 |
| STORAGE(9, P4) | 8.072504 | 0.000000 |
| STORAGE(10, P1) | 5.977499 | 0.000000 |
| STORAGE(10, P2) | 5.963749 | 0.000000 |
| STORAGE(10, P3) | 4.986249 | 0.000000 |
| STORAGE(10, P4) | 8.072504 | 0.000000 |
| STORAGE(11, P1) | 4.977499 | 0.000000 |
| STORAGE(11, P2) | 4.963749 | -0.2585229E-08 |
| STORAGE(11, P3) | 4.986249 | -0.7912372E-08 |
| STORAGE(11, P4) | 8.072504 | 0.000000 |
| STORAGE(12, P1) | 3.977499 | 0.000000 |
| STORAGE(12, P2) | 3.963749 | 0.000000 |
| STORAGE(12, P3) | 4.986249 | 0.000000 |

| | | |
|---|---|---|
| STORAGE(12, P4) | 8.072504 | 0.000000 |
| OUTPUT(0, P1) | 3.000000 | 0.000000 |
| OUTPUT(0, P2) | 4.000000 | 0.000000 |
| OUTPUT(0, P3) | 4.000000 | 0.000000 |
| OUTPUT(0, P4) | 7.000000 | 0.000000 |
| OUTPUT(1, P1) | 3.000000 | 0.000000 |
| OUTPUT(1, P2) | 3.050002 | 0.000000 |
| OUTPUT(1, P3) | 3.050002 | 0.000000 |
| OUTPUT(1, P4) | 7.000000 | 0.000000 |
| OUTPUT(2, P1) | 0.000000 | 0.000000 |
| OUTPUT(2, P2) | 2.570300 | 0.000000 |
| OUTPUT(2, P3) | 4.000000 | 0.000000 |
| OUTPUT(2, P4) | 7.000000 | 0.000000 |
| OUTPUT(3, P1) | 2.000000 | 0.000000 |
| OUTPUT(3, P2) | 2.379697 | 0.000000 |
| OUTPUT(3, P3) | 4.000000 | 0.000000 |
| OUTPUT(3, P4) | 7.000000 | 0.000000 |
| OUTPUT(4, P1) | 3.000000 | 0.000000 |
| OUTPUT(4, P2) | 3.000000 | 0.000000 |
| OUTPUT(4, P3) | 4.000000 | 0.000000 |
| OUTPUT(4, P4) | 7.000000 | 0.000000 |
| OUTPUT(5, P1) | 3.000000 | 0.000000 |
| OUTPUT(5, P2) | 4.000000 | 0.000000 |
| OUTPUT(5, P3) | 4.000000 | 0.000000 |
| OUTPUT(5, P4) | 7.000000 | 0.000000 |
| OUTPUT(6, P1) | 3.000000 | 0.000000 |
| OUTPUT(6, P2) | 4.000000 | 0.000000 |
| OUTPUT(6, P3) | 4.000000 | 0.000000 |
| OUTPUT(6, P4) | 7.000000 | 0.000000 |
| OUTPUT(7, P1) | 3.000000 | 0.000000 |
| OUTPUT(7, P2) | 4.000000 | 0.000000 |
| OUTPUT(7, P3) | 4.000000 | 0.000000 |
| OUTPUT(7, P4) | 7.000000 | 0.000000 |
| OUTPUT(8, P1) | 1.022501 | -0.1675176E-07 |
| OUTPUT(8, P2) | 4.000000 | 0.000000 |
| OUTPUT(8, P3) | 4.000000 | 0.000000 |
| OUTPUT(8, P4) | 7.000000 | 0.000000 |
| OUTPUT(9, P1) | 3.000000 | 0.000000 |
| OUTPUT(9, P2) | 3.036251 | 0.000000 |
| OUTPUT(9, P3) | 4.000000 | 0.000000 |
| OUTPUT(9, P4) | 7.000000 | 0.000000 |
| OUTPUT(10, P1) | 3.000000 | 0.000000 |

| | | |
|---|---|---|
| OUTPUT (10, P2) | 4.000000 | 0.000000 |
| OUTPUT (10, P3) | 4.000000 | 0.000000 |
| OUTPUT (10, P4) | 7.000000 | 0.000000 |
| OUTPUT (11, P1) | 3.000000 | 0.000000 |
| OUTPUT (11, P2) | 4.000000 | 0.000000 |
| OUTPUT (11, P3) | 4.000000 | 0.000000 |
| OUTPUT (11, P4) | 7.000000 | 0.000000 |
| OUTPUT (12, P1) | 3.000000 | 0.000000 |
| OUTPUT (12, P2) | 4.000000 | 0.000000 |
| OUTPUT (12, P3) | 4.000000 | 0.000000 |
| OUTPUT (12, P4) | 7.000000 | 0.000000 |
| BENEFIT (0, B1) | 1.100000 | 0.000000 |
| | ⋮ | |
| BENEFIT (12, B5) | 0.000000 | 0.000000 |

**同样，求解结果也传送到** Excel **数据表中，如图** 13-6 **所示。**

| 运行期 | 水库流出水量(output) | | | | 运行期 | 水库存储水量(storage) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | | P1 | P2 | P3 | P4 |
| 0 | 3 | 4 | 4 | 7 | 0 | 10 | 10 | 10 | 15 |
| 1 | 3 | 3.05 | 3.05 | 7 | 1 | 9 | 9 | 10 | 15 |
| 2 | 0 | 2.5703 | 4 | 7 | 2 | 8 | 8.95 | 10 | 14.05 |
| 3 | 2 | 2.3797 | 4 | 7 | 3 | 10 | 9.3797 | 8.5703 | 11.05 |
| 4 | 3 | 3 | 4 | 7 | 4 | 10 | 10 | 6.95 | 10.05 |
| 5 | 3 | 4 | 4 | 7 | 5 | 9 | 10 | 5.95 | 10.05 |
| 6 | 3 | 4 | 4 | 7 | 6 | 8 | 9 | 5.95 | 10.05 |
| 7 | 3 | 4 | 4 | 7 | 7 | 7 | 8 | 5.95 | 10.05 |
| 8 | 1.0225 | 4 | 4 | 7 | 8 | 6 | 7 | 5.95 | 10.05 |
| 9 | 3 | 3.03625 | 4 | 7 | 9 | 6.9775 | 6 | 5.95 | 8.0725 |
| 10 | 3 | 4 | 4 | 7 | 10 | 5.9775 | 5.96375 | 4.98625 | 8.0725 |
| 11 | 3 | 4 | 4 | 7 | 11 | 4.9775 | 4.96375 | 4.98625 | 8.0725 |
| 12 | 3 | 4 | 4 | 7 | 12 | 3.9775 | 3.96375 | 4.98625 | 8.0725 |

**图** 13-6　**求解后的结果表**

## 13.9　多目标土地规划问题

### 13.9.1　问题描述

**某国家的土地管理机构要对** 100 $km^2$**的公有土地制定规划。根据土地多种用途的总政策，该机构决定把这片土地分成三个区，即野生动物保护区、公共游乐区和商业性林区。政府拥有土地权，并预算每年最多支出** 90 000 **美元管理这片土地，而野生动物保护区、公共游乐区和商业性林区三种土地用途的年费用分别为** 1 000 **美元/**$km^2$**、**4 000 **美元/**$km^2$**和** 5 000 **美元/**$km^2$**。商业性林区每年给该机构带来的收入** 7 000 **美元/**$km^2$**。假定这片土地对三种用途都是适用的，应该如何对这片土地进行规划。**

**这是一个多目标问题，解决多目标问题的常用方法是将其转化为单目标问题进行求解，而**

分目标的指数加权乘积就是这种方法的常用形式。对于本例来说,可以建立如下函数式:

$$Z = F_1(X_1)^a F_2(X_2)^b F_3(X_3)^c$$

式中:$X_1$、$X_2$、$X_3$——野生动物保护区、公共游乐区和商业性林区三种用途土地的面积,$km^2$;

$F_1(X_1)$、$F_2(X_2)$、$F_3(X_3)$——在各种不同的 $X_1$、$X_2$和 $X_3$下估计的价值;

a、b、c——反映目标相对重要性的权重,一般要求 $a + b + c = 1$。

只要能提出价值函数并确定权重,本例就可以用下列最优化模型描述:

$$\max Z = [F_1(X_1)]^a [F_2(X_2)]^b [F_3(X_3)]^c$$

s.t.

$$X_1 + X_2 + X_3 = 100$$

$$1\,000X_1 + 4\,000X_2 - 2\,000X_3 \leqslant 90\,000$$

$$X_1 \geqslant 0$$

$$X_2 \geqslant 0$$

$$X_3 \geqslant 0$$

根据决策者的讨论,表 13-5 中列出每个目标可能的价值和权重,要求决策者最终确定每种土地用途的面积以使总目标函数最大化。

表 13-5 数据表

| 土地用途 | 年费用(美元/ $km^2$) | 目标 X($km^2$) | 目标 F(价值) | 权重 |
|---|---|---|---|---|
| 野生动物保护区 | 1 000 | 0 | 1 | 0.2 |
| | | 20 | 2 | |
| | | 40 | 5 | |
| | | 60 | 8 | |
| | | 80 | 9 | |
| | | 100 | 10 | |
| 公共游乐区 | 4 000 | 0 | 1 | 0.5 |
| | | 20 | 6 | |
| | | 40 | 8 | |
| | | 60 | 10 | |
| | | 80 | 10 | |
| | | 100 | 10 | |
| 商业性林区 | 5 000 - 7 000 = - 2 000 | 0 | 1 | 0.3 |
| | | 20 | 3 | |
| | | 40 | 5 | |
| | | 60 | 7 | |
| | | 80 | 9 | |
| | | 100 | 10 | |

## 13.9.2 LINGO 模型

第一种方法是将该模型考虑为离散的问题,即每种用途的土地面积严格取 0、20、40、60、80 和 100,则可用以下模型直接求解:

```
SETS:
  !用地类型,WEIGHT 权重,EXPEND 年费,Z 价值;
  OBJ/1..3/ :WEIGHT,EXPEND,Z;
  !面积级别:0 20 40 60 80 100;
  KM2/1..6/;
  !规划方案;
  ARCS(OBJ,KM2) :VALUE,SELECTION,STATUS;
ENDSETS
DATA:
  WEIGHT = 0.2 0.5 0.3;
  EXPEND = 1000 4000 -2000;
  VALUE =  1 2 5 8 9 10
           1 6 8 10 10 10
           1 3 5 7 9 10;
  STATUS = 0 20 40 60 80 100
           0 20 40 60 80 100
           0 20 40 60 80 100;
    ENDDATA
MAX = Z(1) * Z(2) * Z(3);
@FOR(OBJ(I):Z(I) = @SUM(ARCS(I,J) :(VALUE(I,J) * SELECTION(I,J))^WEIGHT(I)));
@FOR(ARCS:@BIN(SELECTION));
@FOR(OBJ(I):@SUM(ARCS(I,K) :SELECTION(I,K)) = 1);
@SUM(ARCS(I,J) :STATUS(I,J) * SELECTION(I,J)) = 100;
@SUM(ARCS(I,J) :(STATUS(I,J) * SELECTION(I,J)) * EXPEND(I)) < = 90000;
```

第二种方法是取目标函数的自然对数,即:

$$\max Z' = \ln(Z) = 0.2\ln F_1(X_1) + 0.5\ln F_2(X_2) + 0.3\ln F_3(X_3)$$

其中,函数 $\ln F_j(X_j)$ 及其线性逼近如图 13-7 所示。

利用分段加权法,针对每个函数定义变量 $W_{jk}$,线性近似表达式为:

$$\ln F_1 = 2.08W_{11} + 2.30W_{12}$$

$$\ln F_2 = 1.79W_{21} + 2.30W_{22} + 2.30W_{23}$$

$$\ln F_3 = 1.10W_{31} + 1.95W_{32} + 2.30W_{33}$$

则目标函数相应改变为:

$$\max Z' = 0.416W_{11} + 0.46W_{12} + 0.895W_{21} + 1.15W_{22} + 1.15W_{23} + 0.33W_{31} + 0.585W_{32} + 0.69W_{33}$$

原决策变量变为:

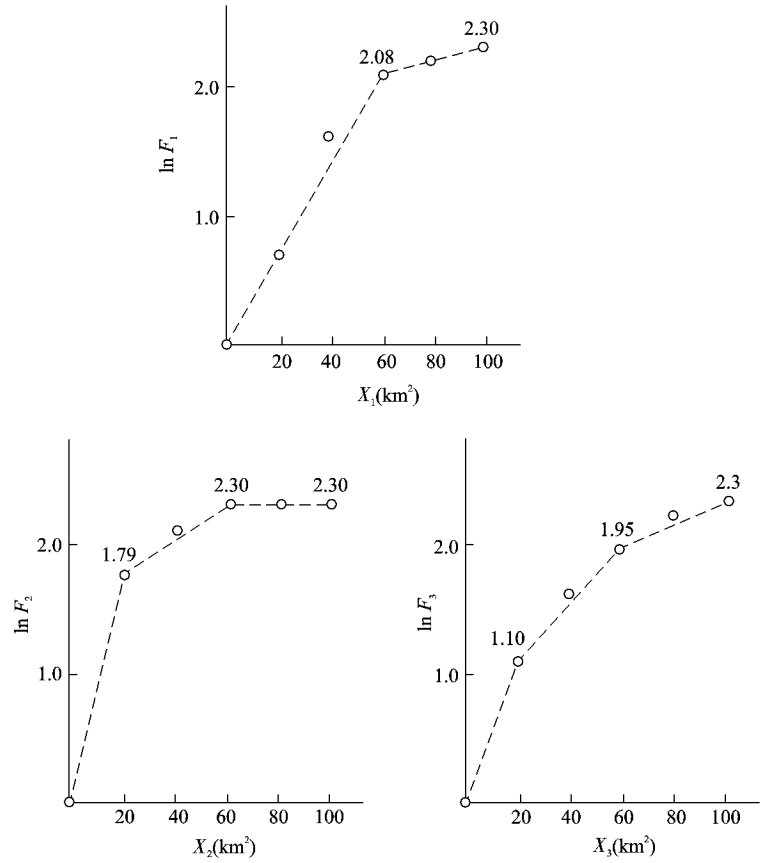



**图 13-7　函数 $\ln F_j(X_j)$ 及其线形逼近图**

$X_1 = 60\ W_{11} + 100\ W_{12}$

$X_2 = 20\ W_{21} + 60\ W_{22} + 100\ W_{23}$

$X_3 = 20\ W_{31} + 60\ W_{32} + 100\ W_{33}$

**经过以上替换,原多目标模型转化为下列线性规划问题：**

$$\max Z' = 0.416W_{11} + 0.46W_{12} + 0.895W_{21} + 1.15W_{22} + 1.15W_{23} + 0.33W_{31} + 0.585W_{32} + 0.69W_{33}$$

s.t.

$60W_{11} + 100\ W_{12} + 20\ W_{21} + 60\ W_{22} + 100\ W_{23} + 20\ W_{31} + 60\ W_{32} + 100\ W_{33} = 100$

$60W_{11} + 100W_{12} + 80W_{21} + 240W_{22} + 400W_{23} - 40\ W_{31} - 120\ W_{32} - 200\ W_{33} \leqslant 90$

$W_{10} + W_{11} + W_{12} = 1$

$W_{20} + W_{21} + W_{22} + W_{23} = 1$

$W_{30} + W_{31} + W_{32} + W_{33} = 1$

$W_{ij} \geqslant 0\ (0 \leqslant i \leqslant 3, 0 \leqslant j \leqslant 3)$

**根据上述改变,建立如下 LINGO 模型：**

SETS：

```
  S /A B C/ :X;
  T /A B C D/;
  U(S,T) :W, OBJ, ST1, ST2, XW, INCLUDE;
ENDSETS
DATA:
  OBJ = 0 0.416 0.46 0
        0 0.895 1.15 1.15
        0 0.33 0.585 0.69;
  ST1 = 0 60 100 0
        0 20 60 100
        0 20 60 100;
  ST2 = 0 60 100 0
        0 80 240 400
        0 -40 -120 -200;
     XW = 0 60 100 0
          0 20 60 100
          0 20 60 100;
   ENDDATA
MAX = @EXP(@SUM(U :W * OBJ * INCLUDE));
@FOR(U(I,J) :INCLUDE(I,J) = @IF(OBJ(I,J) #EQ# 0,0,1));
@SUM(U :W * ST1 * INCLUDE) = 100;
@SUM(U :W * ST2 * INCLUDE) <= 90;
   @FOR(S(I) :@SUM(T(J) :W(I,J) * INCLUDE(I,J)) + W(I,1) = 1);
@FOR(S(I) :X(I) = @SUM(T(J) :XW(I,J) * W(I,J) * INCLUDE(I,J)));
```

## 13.9.3 求解结果

**求解第一种方法的模型,得到如下结果:**

```
Local optimal solution found at iteration:          1551
Objective value:                                5.477226

                    Variable           Value        Reduced Cost
                   WEIGHT(1)       0.2000000            0.000000
                   WEIGHT(2)       0.5000000            0.000000
                   WEIGHT(3)       0.3000000            0.000000
                   EXPEND(1)        1000.000            0.000000
                   EXPEND(2)        4000.000            0.000000
                   EXPEND(3)       -2000.000            0.000000
                        Z(1)        1.379730           -2656413
                        Z(2)        2.449490            0.000000
                        Z(3)        1.620657            0.000000
                  VALUE(1, 1)       1.000000            0.000000
                              ⋮
```

| | | |
|---|---|---|
| VALUE(3, 6) | 10.00000 | 0.000000 |
| SELECTION(1, 1) | 0.000000 | 0.000000 |
| SELECTION(1, 2) | 0.000000 | 0.000000 |
| SELECTION(1, 3) | 1.000000 | 0.000000 |
| SELECTION(1, 4) | 0.000000 | 0.000000 |
| SELECTION(1, 5) | 0.000000 | 0.000000 |
| SELECTION(1, 6) | 0.000000 | 0.000000 |
| SELECTION(2, 1) | 0.000000 | 0.000000 |
| SELECTION(2, 2) | 1.000000 | 0.000000 |
| SELECTION(2, 3) | 0.000000 | 0.000000 |
| SELECTION(2, 4) | 0.000000 | 0.000000 |
| SELECTION(2, 5) | 0.000000 | 0.000000 |
| SELECTION(2, 6) | 0.000000 | 0.000000 |
| SELECTION(3, 1) | 0.000000 | 0.000000 |
| SELECTION(3, 2) | 0.000000 | 0.000000 |
| SELECTION(3, 3) | 1.000000 | 0.000000 |
| SELECTION(3, 4) | 0.000000 | 0.000000 |
| SELECTION(3, 5) | 0.000000 | 0.000000 |
| SELECTION(3, 6) | 0.000000 | 0.000000 |
| STATUS(1, 1) | 0.000000 | 0.000000 |
| ⋮ | | |
| STATUS(3, 6) | 100.0000 | 0.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 5.477226 | 1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |
| 6 | 0.000000 | 0.000000 |
| 7 | 0.000000 | 0.000000 |
| 8 | 0.000000 | 0.000000 |
| 9 | 50000.00 | 0.000000 |

**可见,野生动物保护区、公共游乐区和商业性林区的面积分别为** 40 $km^2$、20 $km^2$ **和** 40 $km^2$。**求解第二种方法的模型,得到如下结果:**

Local optimal solution found at iteration: 17
Objective value: 5.150732

| Variable | Value | Reduced Cost |
|---|---|---|
| X(A) | 56.66667 | 0.000000 |
| X(B) | 20.00000 | 0.000000 |
| X(C) | 23.33333 | 0.000000 |
| W(A, A) | 0.5555556E-01 | 0.000000 |

| | | |
|---|---|---|
| W (A, B) | 0.9444444 | 0.000000 |
| W (A, C) | 0.000000 | 1.201838 |
| W (A, D) | 0.000000 | 0.000000 |
| W (B, A) | 0.000000 | 3.838155 |
| W (B, B) | 1.000000 | 0.000000 |
| W (B, C) | 0.000000 | 0.2300673 |
| W (B, D) | 0.000000 | 1.773570 |
| W (C, A) | 0.000000 | 1.043023 |
| W (C, B) | 0.9166667 | 0.000000 |
| W (C, C) | 0.8333333E-01 | 0.000000 |
| W (C, D) | 0.000000 | 0.7726097 |
| OBJ (A, A) | 0.000000 | 0.000000 |
| | ⋮ | |
| ST1 (A, A) | 0.000000 | 0.000000 |
| | ⋮ | |
| ST2 (C, D) | -200.0000 | 0.000000 |
| XW (A, A) | 0.000000 | 0.000000 |
| | ⋮ | |
| INCLUDE (A, A) | 0.000000 | 0.000000 |
| INCLUDE (A, B) | 1.000000 | 0.000000 |
| INCLUDE (A, C) | 1.000000 | 0.000000 |
| INCLUDE (A, D) | 0.000000 | 0.000000 |
| INCLUDE (B, A) | 0.000000 | 0.000000 |
| INCLUDE (B, B) | 1.000000 | 0.000000 |
| INCLUDE (B, C) | 1.000000 | 0.000000 |
| INCLUDE (B, D) | 1.000000 | 0.000000 |
| INCLUDE (C, A) | 0.000000 | 0.000000 |
| INCLUDE (C, B) | 1.000000 | 0.000000 |
| INCLUDE (C, C) | 1.000000 | 0.000000 |
| INCLUDE (C, D) | 1.000000 | 0.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 5.150732 | 1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.9685980E-04 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |
| 6 | 0.000000 | 0.000000 |
| 7 | 0.000000 | 0.5026493E-03 |
| 8 | 0.000000 | 0.000000 |
| 9 | 0.000000 | 0.000000 |
| 10 | 0.000000 | 0.000000 |
| 11 | 0.000000 | 0.5741911E-04 |

```
12          0.000000          0.1491410E-05
13          0.000000           0.000000
14          0.000000          0.3475314E-01
15          0.000000          0.9586104E-03
16          0.000000           0.000000
17          0.000000           3.838155
18          0.000000           1.043023
19          0.000000           0.000000
20          0.000000           0.000000
21          0.000000           0.000000
```

通过对这两种模型的求解可以发现，线性模型比非线性模型的求解速度快很多。但是，对于 LINGO 来说，如果能够找到合适的数学表达式表述最优化问题，则无需考虑模型的人工简化问题。从求解的结果来看，第二种方法所得结果与参考文献中的相同，而第一种方法的结果要优于后者。

## 13.10 用地规划问题

### 13.10.1 问题描述

某城市的规划部门计划扩大公众参与娱乐活动的机会，初步研究决定增加三个新设施，即游泳池、小公园和网球场。在这座城市中，有四个可能的场地可用来建设这些设施。由于基建要求、场地条件和征得土地的情况有所不同，因此在不同场地上建设各种设施的费用也不同，如表 13-6 所示。规划部门希望确定建设这三个设施的合理地点以最大限度地降低费用。

表 13-6 在四个不同场地建设娱乐设施的费用

| 设施 | 场地费用 ($10^3$元/年) | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 游泳池 | 20 | 40 | 15 | 30 |
| 公园 | 5 | 10 | 13 | 6 |
| 网球场 | 8 | 35 | 16 | 28 |

根据题意，建立如下最优化模型：

$$\min\ z = \sum_{j=1}^{3}\sum_{i=1}^{4} c_{ij}x_{ij}$$

s.t.

$$\sum_{j=1}^{4} x_{ij} = 1$$

$$\sum_{i=1}^{3} x_{ij} = 1$$

$x_{ij} = 1$ 或 $x_{ij} = 0$

式中：$c_{ij}$——在 j 场地建设 i 设施的费用，$10^3$元/年；

$x_{ij}$——表示是否在 j 场地建设 i 设施。

## 13.10.2 LINGO 模型

根据题意，建立如下 LINGO 模型：

```
SETS:
  SERVICE/POOL,PARK,TENNIS/;
  PLACES /P1,P2,P3,P4/;
  PAIRS (SERVICE, PLACES) :COST, MATCH;
ENDSETS
DATA:
  COST = 20 40 15 30
         5 10 13 6
         8 35 16 28;
ENDDATA

MIN = @SUM(PAIRS(I, J):COST(I, J) * MATCH(I, J));
@FOR(SERVICE(I):
  @SUM(PAIRS(I, K) :MATCH(I, K)) = 1);
@FOR(SERVICE(I):
  @SUM(PAIRS(J, I) :MATCH(J, K)) = 1);
@FOR(PAIRS(I, J):@BIN(MATCH(I, J)));
```

## 13.10.3 求解结果

求解上述模型，得到下列结果：

```
Global optimal solution found at iteration:          0
Objective value:                                29.00000

                     Variable           Value        Reduced Cost
                   COST(POOL, P1)       20.00000        0.000000
                   COST(POOL, P2)       40.00000        0.000000
                   COST(POOL, P3)       15.00000        0.000000
                   COST(POOL, P4)       30.00000        0.000000
                   COST(PARK, P1)       5.000000        0.000000
                   COST(PARK, P2)       10.00000        0.000000
                   COST(PARK, P3)       13.00000        0.000000
                   COST(PARK, P4)       6.000000        0.000000
                  COST(TENNIS, P1)      8.000000        0.000000
```

| | | |
|---|---|---|
| COST(TENNIS, P2) | 35.00000 | 0.000000 |
| COST(TENNIS, P3) | 16.00000 | 0.000000 |
| COST(TENNIS, P4) | 28.00000 | 0.000000 |
| MATCH(POOL, P1) | 0.000000 | 20.00000 |
| MATCH(POOL, P2) | 0.000000 | 40.00000 |
| MATCH(POOL, P3) | 1.000000 | 15.00000 |
| MATCH(POOL, P4) | 0.000000 | 30.00000 |
| MATCH(PARK, P1) | 0.000000 | 5.000000 |
| MATCH(PARK, P2) | 0.000000 | 10.00000 |
| MATCH(PARK, P3) | 0.000000 | 13.00000 |
| MATCH(PARK, P4) | 1.000000 | 6.000000 |
| MATCH(TENNIS, P1) | 1.000000 | 8.000000 |
| MATCH(TENNIS, P2) | 0.000000 | 35.00000 |
| MATCH(TENNIS, P3) | 0.000000 | 16.00000 |
| MATCH(TENNIS, P4) | 0.000000 | 28.00000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 29.00000 | -1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |
| 6 | 1.000000 | 0.000000 |
| 7 | 0.000000 | 0.000000 |
| 8 | 0.000000 | 0.000000 |

## 13.11 大气污染物排放的控制问题

### 13.11.1 问题描述

**在一个小区内有三个排放总悬浮颗粒物(TSP)的点源,其中两个是燃煤发电厂,另一个是水泥厂。发电厂每烧** 1 t **煤排放** 95 kg **总悬浮颗粒物,而水泥厂每生产** 1 t **水泥排放** 85 kg **总悬浮颗粒物。两个发电厂燃煤量分别为** 400 000 t/a **和** 300 000 t/a**,水泥厂产量为** 250 000 t/a**。当前,这三个总悬浮颗粒物排放源都没有控制措施。现在,每个点源可以选择的控制方法的去除效率和费用如表** 13-7 **所示。此问题的目标是以最小的费用把水泥厂和两个发电厂的总悬浮颗粒物总排放量削减** 80%**。**

表 13-7　数据表

| 点源 | 控制方法 | 去除效率 (%) | 未去除效率 (%) | 费用 (元/t) | 燃煤量 (t/a) | TSP 排放量 (kg/t) | 排放总量 (kg/a) |
|---|---|---|---|---|---|---|---|
| 发电厂 1 | 隔板 | 59 | 41 | 1.00 | 400 000 | 95 | 38 000 000 |
| | 喷雾 | 94 | 6 | 2.00 | | | |
| | 静电 | 97 | 3 | 2.80 | | | |
| 发电厂 2 | 隔板 | 59 | 41 | 1.40 | 300 000 | 95 | 28 500 000 |
| | 喷雾 | 94 | 6 | 2.20 | | | |
| | 静电 | 97 | 3 | 3.00 | | | |
| 水泥厂 | 隔板 | 59 | 41 | 1.10 | 250 000 | 85 | 21 250 000 |
| | 多级 | 74 | 26 | 1.20 | | | |
| | 长锥 | 84 | 16 | 1.50 | | | |
| | 喷雾 | 94 | 6 | 3.00 | | | |

现在排放量总计为 87 750 000 kg/a。由于要求总排放量削减 80%，则最大允许排放量应为 17 550 000 kg/a。

根据题意，建立如下数学模型：

$$\min Z = 1.0X_{11} + 2.0X_{14} + 2.8X_{15} + 1.4X_{21} + 2.2X_{24} + 3.0X_{25} + 1.1X_{31} + 1.2X_{32} + 1.5X_{33} + 3.0X_{34}$$

s.t.

$$X_{10} + X_{11} + X_{14} + X_{15} = 400\ 000$$

$$X_{20} + X_{21} + X_{24} + X_{25} = 300\ 000$$

$$X_{30} + X_{31} + X_{32} + X_{33} + X_{34} = 250\ 000$$

$$95X_{10} + 39.0X_{11} + 5.7X_{14} + 2.9X_{15} + 95X_{20} + 39.0X_{21} + 5.7X_{24} + 2.9X_{25} + 85X_{30} + 34.9X_{31} + 22.1X_{32} + 13.6X_{33} + 5.1X_{34} \leqslant 17\ 550\ 000$$

$$X_{ij} \geqslant 100$$

式中：i——污染源，1 表示发电厂 1，2 表示发电厂 2，3 表示水泥厂；

j——控制方法，0 表示不控制，1 表示隔板沉淀槽，2 表示多级除尘器，3 表示长锥除尘器，4 表示喷雾洗涤器，5 表示静电除尘器；

$X_{ij}$——点源 i 采用控制方法 j 的生产量，t/a。

## 13.11.2　LINGO 模型

根据题意，建立如下 LINGO 模型：

```
SETS:
  POLUTION/ELEC1,ELEC2,CEMENT/:COAL,DISCHARGE;
  CONTROLS/CON0..CON5/:EFFECT;
  ARCS(POLUTION,CONTROLS):PRODUCTION,FARE,INCLUDE;
ENDSETS
```

```
DATA:
  COAL = 400000,300000,250000;
  DISCHARGE = 95,95,85;
  EFFECT = 0,0.59,0.74,0.84,0.94,0.97;
  FARE = 0 1.00 0 0 2.00 2.80
         0 1.40 0 0 2.20 3.00
         0 1.00 1.20 1.50 3.00 0;
ENDDATA

MIN = @SUM(ARCS(I,J) :FARE(I,J) * PRODUCTION(I,J) * INCLUDE(I,J));

@SUM(POLUTION(I) :@SUM(CONTROLS(J) :DISCHARGE(I) * (1 - EFFECT(J)) * PRODUCTION(I,
J) * INCLUDE(I,J))) < = @SUM(POLUTION(I) :COAL(I) * DISCHARGE(I)) * 0.2;
@FOR(POLUTION(I) :@SUM(CONTROLS(J) :PRODUCTION(I,J) * INCLUDE(I,J)) = COAL(I));
@FOR(ARCS(I,J) :INCLUDE(I,J) = @IF(FARE(I,J) #EQ# 0,0,1));
@FOR(ARCS:@GIN(PRODUCTION));
```

## 13.11.3 求解结果

**求解上述模型,得到下列结果:**

```
Global optimal solution found at iteration:          11
Objective value:                                1518346

                         Variable           Value        Reduced Cost
                       COAL(ELEC1)        400000.0            0.000000
                       COAL(ELEC2)        300000.0            0.000000
                     COAL(CEMENT)         250000.0            0.000000
                  DISCHARGE(ELEC1)        95.00000            0.000000
                  DISCHARGE(ELEC2)        95.00000            0.000000
                 DISCHARGE(CEMENT)        85.00000            0.000000
                       EFFECT(CON0)        0.000000            0.000000
                       EFFECT(CON1)        0.5900000           0.000000
                       EFFECT(CON2)        0.7400000           0.000000
                       EFFECT(CON3)        0.8400000           0.000000
                       EFFECT(CON4)        0.9400000           0.000000
                       EFFECT(CON5)        0.9700000           0.000000
             PRODUCTION(ELEC1, CON0)       0.000000            0.000000
             PRODUCTION(ELEC1, CON1)       241654.0            1.000000
             PRODUCTION(ELEC1, CON2)       0.000000            0.000000
             PRODUCTION(ELEC1, CON3)       0.000000            0.000000
             PRODUCTION(ELEC1, CON4)       158346.0            2.000000
```

| | | |
|---|---|---|
| PRODUCTION(ELEC1, CON5) | 0.000000 | 2.800000 |
| PRODUCTION(ELEC2, CON0) | 0.000000 | 0.000000 |
| PRODUCTION(ELEC2, CON1) | 0.000000 | 1.400000 |
| PRODUCTION(ELEC2, CON2) | 0.000000 | 0.000000 |
| PRODUCTION(ELEC2, CON3) | 0.000000 | 0.000000 |
| PRODUCTION(ELEC2, CON4) | 300000.0 | 2.200000 |
| PRODUCTION(ELEC2, CON5) | 0.000000 | 3.000000 |
| PRODUCTION(CEMENT, CON0) | 0.000000 | 0.000000 |
| PRODUCTION(CEMENT, CON1) | 0.000000 | 1.000000 |
| PRODUCTION(CEMENT, CON2) | 250000.0 | 1.200000 |
| PRODUCTION(CEMENT, CON3) | 0.000000 | 1.500000 |
| PRODUCTION(CEMENT, CON4) | 0.000000 | 3.000000 |
| PRODUCTION(CEMENT, CON5) | 0.000000 | 0.000000 |
| FARE(ELEC1, CON0) | 0.000000 | 0.000000 |
| FARE(ELEC1, CON1) | 1.000000 | 0.000000 |
| FARE(ELEC1, CON2) | 0.000000 | 0.000000 |
| FARE(ELEC1, CON3) | 0.000000 | 0.000000 |
| FARE(ELEC1, CON4) | 2.000000 | 0.000000 |
| FARE(ELEC1, CON5) | 2.800000 | 0.000000 |
| FARE(ELEC2, CON0) | 0.000000 | 0.000000 |
| FARE(ELEC2, CON1) | 1.400000 | 0.000000 |
| FARE(ELEC2, CON2) | 0.000000 | 0.000000 |
| FARE(ELEC2, CON3) | 0.000000 | 0.000000 |
| FARE(ELEC2, CON4) | 2.200000 | 0.000000 |
| FARE(ELEC2, CON5) | 3.000000 | 0.000000 |
| FARE(CEMENT, CON0) | 0.000000 | 0.000000 |
| FARE(CEMENT, CON1) | 1.000000 | 0.000000 |
| FARE(CEMENT, CON2) | 1.200000 | 0.000000 |
| FARE(CEMENT, CON3) | 1.500000 | 0.000000 |
| FARE(CEMENT, CON4) | 3.000000 | 0.000000 |
| FARE(CEMENT, CON5) | 0.000000 | 0.000000 |
| INCLUDE(ELEC1, CON0) | 0.000000 | 0.000000 |
| INCLUDE(ELEC1, CON1) | 1.000000 | 0.000000 |
| INCLUDE(ELEC1, CON2) | 0.000000 | 0.000000 |
| INCLUDE(ELEC1, CON3) | 0.000000 | 0.000000 |
| INCLUDE(ELEC1, CON4) | 1.000000 | 0.000000 |
| INCLUDE(ELEC1, CON5) | 1.000000 | 0000000 |
| INCLUDE(ELEC2, CON0) | 0.000000 | 0.000000 |
| INCLUDE(ELEC2, CON1) | 1.000000 | 0.000000 |
| INCLUDE(ELEC2, CON2) | 0.000000 | 0.000000 |
| INCLUDE(ELEC2, CON3) | 0.000000 | 0.000000 |
| INCLUDE(ELEC2, CON4) | 1.000000 | 0.000000 |

| | | |
|---|---|---|
| INCLUDE(ELEC2, CON5) | 1.000000 | 0.000000 |
| INCLUDE(CEMENT, CON0) | 0.000000 | 0.000000 |
| INCLUDE(CEMENT, CON1) | 1.000000 | 0.000000 |
| INCLUDE(CEMENT, CON2) | 1.000000 | 0.000000 |
| INCLUDE(CEMENT, CON3) | 1.000000 | 0.000000 |
| INCLUDE(CEMENT, CON4) | 1.000000 | 0.000000 |
| INCLUDE(CEMENT, CON5) | 0.000000 | 0.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 1518346 | -1.000000 |
| 2 | 4.500000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |
| 6 | 0.000000 | 0.000000 |
| 7 | 0.000000 | -241654.0 |
| 8 | 0.000000 | 0.000000 |
| 9 | 0.000000 | 0.000000 |
| 10 | 0.000000 | -316692.0 |
| 11 | 0.000000 | 0.000000 |
| 12 | 0.000000 | 0.000000 |
| 13 | 0.000000 | 0.000000 |
| 14 | 0.000000 | 0.000000 |
| 15 | 0.000000 | 0.000000 |
| 16 | 0.000000 | -660000.0 |
| 17 | 0.000000 | 0.000000 |
| 18 | 0.000000 | 0.000000 |
| 19 | 0.000000 | 0.000000 |
| 20 | 0.000000 | -300000.0 |
| 21 | 0.000000 | 0.000000 |
| 22 | 0.000000 | 0.000000 |
| 23 | 0.000000 | 0.000000 |

## 13.12 固体废弃物处置问题

### 13.12.1 问题描述

**有两个城市要建设一个区域固体废物处理系统。城市** 1 **有** 40 000 **人,固体废弃物量是** 700 t/**周 城市** 2 **有** 65 000 **人,固体废弃物量是** 1 200 t/**周。现在有三种可能的方案,即焚烧、排海和掩埋。针对这三种方案有三个处理场地,场地处理系统情况如表** 13-8 **所示。已知运输费**

**用为 0.5 美元/(t·km),试确定最优固体废弃物处理方案,使整个系统的费用最低。**

**表 13-8　固体废弃物处理场地及费用、能力情况**

| 场地 | 方法 | 距城市 1 距离 (km) | 距城市 2 距离 (km) | 固定费用 (美元/周) | 可变费用 (美元/t) | 处理能力 (t/周) |
|---|---|---|---|---|---|---|
| 1 | 焚烧 | 15 | 10 | 3 850 | 12 | 1 000 |
| 2 | 排海 | 5 | 15 | 1 150 | 16 | 500 |
| 3 | 掩埋 | 30 | 25 | 1 920 | 6 | 1 300 |

## 13.12.2　LINGO 模型

**根据题意,建立如下 LINGO 模型:**

```
SETS:
   CITY/A,B/:WASTE;
   PLANT/FIRE,OCEAN,BURY/:MATCH,FIX,CHANGE,ABILITY;
   PLAN(CITY,PLANT):PRODUCE,DISTANCE;
ENDSETS

DATA:
  FIX = 3850 1150 1920;
  CHANGE = 12 16 6;
  ABILITY = 1000 500 1300;
  DISTANCE = 15 5 30
             10 15 25;
WASTE = 700 1200;
ENDDATA

MIN = @SUM(PLANT:MATCH*FIX) + @SUM(PLAN(I,J):
      (PRODUCE(I,J)*CHANGE(J)+PRODUCE(I,J)*DISTANCE(I,J)*0.5));

@FOR(CITY(I):SUM(PLAN(I,J):PRODUCE(I,J)) = WASTE(I));
@FOR(PLANT(J):@SUM(PLAN(I,J):PRODUCE(I,J)) <= MATCH(J)*ABILITY(J));
@FOR(PLANT:@BIN(MATCH));
```

## 13.12.3　求解结果

```
Global optimal solution found at iteration:          27

Objective value:                              41070.00

                    Variable           Value        Reduced Cost
```

| | | |
|---|---|---|
| WASTE (A) | 700.0000 | 0.000000 |
| WASTE (B) | 1200.000 | 0.000000 |
| MATCH (FIRE) | 1.000000 | 2350.000 |
| MATCH (OCEAN) | 1.000000 | -100.0000 |
| MATCH (BURY) | 1.000000 | 1920.000 |
| FIX (FIRE) | 3850.000 | 0.000000 |
| FIX (OCEAN) | 1150.000 | 0.000000 |
| FIX (BURY) | 1920.000 | 0.000000 |
| CHANGE (FIRE) | 12.00000 | 0.000000 |
| CHANGE (OCEAN) | 16.00000 | 0.000000 |
| CHANGE (BURY) | 6.000000 | 0.000000 |
| ABILITY (FIRE) | 1000.000 | 0.000000 |
| ABILITY (OCEAN) | 500.0000 | 0.000000 |
| ABILITY (BURY) | 1300.000 | 0.000000 |
| PRODUCE (A, FIRE) | 169.2308 | 0.000000 |
| PRODUCE (A, OCEAN) | 500.0000 | 0.000000 |
| PRODUCE (A, BURY) | 30.76923 | 0.000000 |
| PRODUCE (B, FIRE) | 830.7692 | 0.000000 |
| PRODUCE (B, OCEAN) | 0.000000 | 7.500000 |
| PRODUCE (B, BURY) | 369.2308 | 0.000000 |
| DISTANCE (A, FIRE) | 15.00000 | 0.000000 |
| DISTANCE (A, OCEAN) | 5.000000 | 0.000000 |
| DISTANCE (A, BURY) | 30.00000 | 0.000000 |
| DISTANCE (B, FIRE) | 10.00000 | 0.000000 |
| DISTANCE (B, OCEAN) | 15.00000 | 0.000000 |
| DISTANCE (B, BURY) | 25.00000 | 0.000000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 41070.00 | -1.000000 |
| 2 | 0.000000 | -21.00000 |
| 3 | 0.000000 | -18.50000 |
| 4 | 0.000000 | 1.500000 |
| 5 | 0.000000 | 2.500000 |
| 6 | 900.0000 | 0.000000 |

## 13.13 肥料贮存问题

### 13.13.1 问题描述

**一个组织决定在 5～8 月间经营混合肥料。根据对当地垃圾和有机废物的分析，能在 5、6、7 和 8 月依次生产 500 t、400 t、300 t 和 300 t 混合肥料。然而这期间并不是每个月的产品都能**

在市场上完全销售。通过市场分析,每个月出售肥料能得到的净收入如表 13-9 所示。在这 4 个月内最大销售量分别是 300 t、600 t、800 t 和 500 t。由于预期销售量与生产量不相匹配,决定建设贮存设施以贮存多余的肥料,在以后需求量高时出售。贮存肥料需要的费用见表 13-10。贮存设施的容量不大于 500 t。现在的问题是每个月应该出售和贮存多少肥料才能获得最大的利润。

表 13-9　肥料销售的净收入

| 销售量 (t) | 净收入 ($10^3$ 美元) | | | |
|---|---|---|---|---|
| | 5 月 | 6 月 | 7 月 | 8 月 |
| 0 | 0 | 0 | 0 | 0 |
| 100 | 1 | 2 | 2 | 1 |
| 200 | 2 | 3 | 3 | 2 |
| 300 | 3 | 4 | 4 | 4 |
| 400 | — | 4 | 5 | 6 |
| 500 | — | 4 | 6 | 7 |
| 600 | — | 4 | 8 | — |
| 700 | — | — | 9 | — |
| 800 | — | — | 9 | — |

表 13-10　肥料贮存的月费用

| t 月开始的贮存量 (t) | 贮存费用 ($10^3$ 美元) | | | |
|---|---|---|---|---|
| | 5 月 | 6 月 | 7 月 | 8 月 |
| 0 | 1 | 1 | 2 | 2 |
| 100 | 3 | 2 | 3 | 3 |
| 200 | 4 | 2 | 3 | 3 |
| 300 | 5 | 3 | 4 | 4 |
| 400 | 7 | 4 | 5 | 5 |
| 500 | 8 | 4 | 5 | 5 |

如图 13-8 所示,本模型是一维动态模型的标准形式。以月为阶段,在第 t 月开始时贮存的肥料量为状态变量。图 13-8 的输出是状态变量和决策变量两者的函数。由于在每个阶段可得到的资源 $S_t$ 由外来的补充而增加,在 t 月可得到用于分配的肥料总量为 $S_t + I_t$。所需贮存设施的质量平衡关系为 $S_{t+1} = S_t + I_t - X_t$,其中 $I_t$ 为第 t 月生产的肥料量,$X_t$ 为第 t 月肥料的销售量。因为在这 4 个月的开始和终了时,肥料贮存量均应为零,所以 $S_1 = 0$,$S_4 + I_4 - X_4 = 0$。

图中:$S_t$ 表示 t 月开始时肥料贮存量;$r_t$ $(S_t, X_t)$ 表示第 t 月的利润;$I_t$ 表示第 t 月生产的肥料量。这样,可以建立如下优化模型:

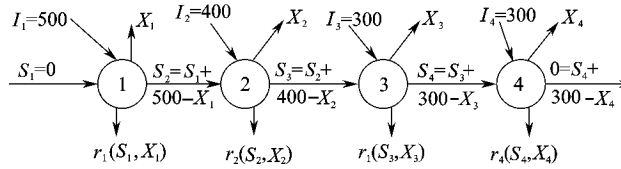



**图 13-8　肥料贮存问题关联图**

$$\max Z = \sum_{t=1}^{4} (B_t - C_t)$$

s.t.

$$S_1 = 0$$

$$S_2 - S_1 + X_1 = 500$$

$$S_3 - S_2 + X_2 = 400$$

$$S_4 - S_3 + X_3 = 300$$

$$- S_4 + X_4 = 300$$

$$S_t \leqslant 500$$

$$X_1 \in \Omega_1 = (0,\cdots,300)$$

$$X_2 \in \Omega_2 = (0,\cdots,600)$$

$$X_3 \in \Omega_3 = (0,\cdots,800)$$

$$X_4 \in \Omega_4 = (0,\cdots,500)$$

**式中：$B_t$表示第 t 月的销售收入；$C_t$表示第 t 月的贮存费用。**

## 13.13.2　LINGO 模型

**根据题意，建立如下 LINGO 模型：**

```
SETS:
  MONTH/MAY,JUNE,JULY,AUGUST/:SALE,STORE,PRODUCE;
  SALEAMOUNT/1..9/:AMOUNTFORSALE;
  STOREAMOUNT/1..6/:AMOUNTFORSTORE;
  MONTHSALE(MONTH,SALEAMOUNT):INCOME,YN1;
  MONTHSTORE(MONTH,STOREAMOUNT):FARE,YN2;
ENDSETS

DATA:
  PRODUCE = 500 400 300 300;
  AMOUNTFORSALE = 0 100 200 300 400 500 600 700 800;
  AMOUNTFORSTORE = 0 100 200 300 400 500;
  INCOME = 0 1 2 3 0 0 0 0 0
           0 2 3 4 4 4 4 0 0
           0 2 3 4 5 6 8 9 9
```

```
          0 1 2 4 6 7 0 0 0;
  FARE = 1 3 4 5 7 8
         1 2 2 3 4 4
         2 3 3 4 5 5
         2 3 3 4 5 5;
ENDDATA

MAX = @SUM(MONTHSALE(I,J):INCOME(I,J) * YN1(I,J) * AMOUNTFORSALE(J))
    - @SUM(MONTHSTORE(I,J):FARE(I,J) * YN2(I,J) * AMOUNTFORSTORE(J));

@FOR(MONTH(I):SALE(I) = @SUM(SALEAMOUNT(J):AMOUNTFORSALE(J) * YN1(I,J)));
@FOR(MONTH(I):STORE(I) = @SUM(STOREAMOUNT(J):AMOUNTFORSTORE(J) * YN2(I,J)));
@FOR(MONTH(I)|I#LE#3:STORE(I+1) - STORE(I) - PRODUCE(I) + SALE(I) = 0);
   SALE(4) - STORE(4) = PRODUCE(4);
@FOR(MONTH(I):@SUM(SALEAMOUNT(J):YN1(I,J)) = 1);
@FOR(MONTH(I):@SUM(STOREAMOUNT(J):YN2(I,J)) = 1);
@FOR(MONTHSALE:@BIN(YN1));
@FOR(MONTHSTORE:@BIN(YN2));
```

## 13.13.3 求解结果

**求解上述模型,得到下列结果：**

```
Global optimal solution found at iteration:          0
Objective value:                              6600.000

                    Variable           Value        Reduced Cost
                  SALE(MAY)          300.0000            0.000000
                 SALE(JUNE)          100.0000            0.000000
                 SALE(JULY)          800.0000            0.000000
               SALE(AUGUST)          300.0000            0.000000
                 STORE(MAY)          0.000000            0.000000
                STORE(JUNE)          200.0000            0.000000
                STORE(JULY)          500.0000            0.000000
              STORE(AUGUST)          0.000000            0.000000
               PRODUCE(MAY)          500.0000            0.000000
              PRODUCE(JUNE)          400.0000            0.000000
              PRODUCE(JULY)          300.0000            0.000000
            PRODUCE(AUGUST)          300.0000            0.000000
          AMOUNTFORSALE(1)           0.000000            0.000000
                                        ⋮
         AMOUNTFORSTORE(6)           500.0000            0.000000
             INCOME(MAY, 1)          0.000000            0.000000
```

| | | |
|---|---|---|
| | ⋮ | |
| INCOME (AUGUST, 9) | 0.000000 | 0.000000 |
| YN1 (MAY, 1) | 0.000000 | 0.000000 |
| YN1 (MAY, 2) | 0.000000 | -100.0000 |
| YN1 (MAY, 3) | 0.000000 | -400.0000 |
| YN1 (MAY, 4) | 1.000000 | -900.0000 |
| YN1 (MAY, 5) | 0.000000 | 0.000000 |
| YN1 (MAY, 6) | 0.000000 | 0.000000 |
| YN1 (MAY, 7) | 0.000000 | 0.000000 |
| YN1 (MAY, 8) | 0.000000 | 0.000000 |
| YN1 (MAY, 9) | 0.000000 | 0.000000 |
| YN1 (JUNE, 1) | 0.000000 | 0.000000 |
| YN1 (JUNE, 2) | 1.000000 | -200.0000 |
| YN1 (JUNE, 3) | 0.000000 | -600.0000 |
| YN1 (JUNE, 4) | 0.000000 | -1200.000 |
| YN1 (JUNE, 5) | 0.000000 | -1600.000 |
| YN1 (JUNE, 6) | 0.000000 | -2000.000 |
| YN1 (JUNE, 7) | 0.000000 | -2400.000 |
| YN1 (JUNE, 8) | 0.000000 | 0.000000 |
| YN1 (JUNE, 9) | 0.000000 | 0.000000 |
| YN1 (JULY, 1) | 0.000000 | 0.000000 |
| YN1 (JULY, 2) | 0.000000 | -200.0000 |
| YN1 (JULY, 3) | 0.000000 | -600.0000 |
| YN1 (JULY, 4) | 0.000000 | -1200.000 |
| YN1 (JULY, 5) | 0.000000 | -2000.000 |
| YN1 (JULY, 6) | 0.000000 | -3000.000 |
| YN1 (JULY, 7) | 0.000000 | -4800.000 |
| YN1 (JULY, 8) | 0.000000 | -6300.000 |
| YN1 (JULY, 9) | 1.000000 | -7200.000 |
| YN1 (AUGUST, 1) | 0.000000 | 0.000000 |
| YN1 (AUGUST, 2) | 0.000000 | -100.0000 |
| YN1 (AUGUST, 3) | 0.000000 | -400.0000 |
| YN1 (AUGUST, 4) | 1.000000 | -1200.000 |
| YN1 (AUGUST, 5) | 0.000000 | -2400.000 |
| YN1 (AUGUST, 6) | 0.000000 | -3500.000 |
| YN1 (AUGUST, 7) | 0.000000 | 0.000000 |
| YN1 (AUGUST, 8) | 0.000000 | 0.000000 |
| YN1 (AUGUST, 9) | 0.000000 | 0.000000 |
| FARE (MAY, 1) | 1.000000 | 0.000000 |
| | ⋮ | |
| FARE (AUGUST, 6) | 5.000000 | 0.000000 |
| YN2 (MAY, 1) | 1.000000 | 0.000000 |

| | | |
|---|---|---|
| YN2 (MAY, 2) | 0.000000 | 300.0000 |
| YN2 (MAY, 3) | 0.000000 | 800.0000 |
| YN2 (MAY, 4) | 0.000000 | 1500.000 |
| YN2 (MAY, 5) | 0.000000 | 2800.000 |
| YN2 (MAY, 6) | 0.000000 | 4000.000 |
| YN2 (JUNE, 1) | 0.000000 | 0.000000 |
| YN2 (JUNE, 2) | 0.000000 | 200.0000 |
| YN2 (JUNE, 3) | 1.000000 | 400.0000 |
| YN2 (JUNE, 4) | 0.000000 | 900.0000 |
| YN2 (JUNE, 5) | 0.000000 | 1600.000 |
| YN2 (JUNE, 6) | 0.000000 | 2000.000 |
| YN2 (JULY, 1) | 0.000000 | 0.000000 |
| YN2 (JULY, 2) | 0.000000 | 300.0000 |
| YN2 (JULY, 3) | 0.000000 | 600.0000 |
| YN2 (JULY, 4) | 0.000000 | 1200.000 |
| YN2 (JULY, 5) | 0.000000 | 2000.000 |
| YN2 (JULY, 6) | 1.000000 | 2500.000 |
| YN2 (AUGUST, 1) | 1.000000 | 0.000000 |
| YN2 (AUGUST, 2) | 0.000000 | 300.0000 |
| YN2 (AUGUST, 3) | 0.000000 | 600.0000 |
| YN2 (AUGUST, 4) | 0.000000 | 1200.000 |
| YN2 (AUGUST, 5) | 0.000000 | 2000.000 |
| YN2 (AUGUST, 6) | 0.000000 | 2500.000 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| 1 | 6600.000 | 1.000000 |
| 2 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 0.000000 |
| 4 | 0.000000 | 0.000000 |
| 5 | 0.000000 | 0.000000 |
| 6 | 0.000000 | 0.000000 |
| 7 | 0.000000 | 0.000000 |
| 8 | 0.000000 | 0.000000 |
| 9 | 0.000000 | 0.000000 |
| 10 | 0.000000 | 0.000000 |
| 11 | 0.000000 | 0.000000 |
| 12 | 0.000000 | 0.000000 |
| 13 | 0.000000 | 0.000000 |
| 14 | 0.000000 | 0.000000 |
| 15 | 0.000000 | 0.000000 |
| 16 | 0.000000 | 0.000000 |
| 17 | 0.000000 | 0.000000 |

| | | |
|---|---|---|
| 18 | 0.000000 | 0.000000 |
| 19 | 0.000000 | 0.000000 |
| 20 | 0.000000 | 0.000000 |
| 21 | 0.000000 | 0.000000 |

## 13.14 农业面源污染控制问题

### 13.14.1 问题描述

围绕某个湖泊有 1 000 $hm^2$ 农田,这片农田适合于种植两种作物。作物 1 的收入为 300 美元/$hm^2$,作物 2 为 150 美元/$hm^2$。不考虑种植费用的经济规模性,种植作物 1 的平均费用为 160 美元/$hm^2$,作物 2 为 50 美元/$hm^2$。但是,种植作物 1 每公顷农药流失量为 0.9 kg/a,种植作物 2 每公顷农药流失量为 0.5 kg/a,并且总农药流失量不允许超过 632.5 kg/a。现在的问题是确定在保护湖泊环境的前提下,使农场主获得最大收益的最佳种植组合。

将问题的决策变量定义为 $X_1$ 和 $X_2$,分别代表作物 1 和作物 2 的种植面积($hm^2$),所以年收益(美元/a)为:

$$Z = (300 - 160)X_1 + (150 - 50)X_2 = 140X_1 + 100X_2$$

这样,最优化模型如下:

$$\max Z = 140X_1 + 100X_2$$

s.t.

$$0.9X_1 + 0.5X_2 \leqslant 632.5$$

$$X_1 + X_2 \leqslant 1\ 000$$

$$X_1 \geqslant 0$$

$$X_2 \geqslant 0$$

### 13.14.2 LINGO 模型

根据题意,建立如下 LINGO 模型:

```
SETS:
  CROP/1 2/:PESTICIDE,COST,GAIN,AREA;
ENDSETS
DATA:
  PESTICIDE = 0.9 0.5;
  COST = 160 50;
  GAIN = 300 150;
  TOTAL_AREA = 1000; TOTAL_PESTICIDE = 632.5;
ENDDATA

[TOTAL_GAIN] MAX = @SUM(CROP(I):(GAIN(I) - COST(I)) * AREA(I));
```

```
@SUM(CROP(I):PESTICIDE(I)*AREA(I))<=TOTAL_PESTICIDE;
@SUM(CROP(I):AREA(I))<=TOTAL_AREA;
```

## 13.14.3 求解结果

**求解上述模型,得到下列结果:**

```
Global optimal solution found at iteration:          0
Objective value:                              113250.0

                    Variable           Value        Reduced Cost
                  TOTAL_AREA        1000.000            0.000000
               TOTAL_PESTICIDE      632.5000            0.000000
                 PESTICIDE(1)      0.9000000           0.000000
                 PESTICIDE(2)      0.5000000           0.000000
                      COST(1)       160.0000           0.000000
                      COST(2)       50.00000           0.000000
                      GAIN(1)       300.0000           0.000000
                      GAIN(2)       150.0000           0.000000
                      AREA(1)       331.2500           0.000000
                      AREA(2)       668.7500           0.000000

                         Row    Slack or Surplus      Dual Price
                   TOTAL_GAIN      113250.0            1.000000
                           2       0.000000            100.0000
                           3       0.000000            50.00000
```

## 13.14.4 应用 Delphi 建立模型界面

**在 Delphi 中编辑程序的界面如图 13-9 所示。**

**"计算"按钮的程序如下:**

```
implementation
uses lingodel;  //引用外部函数单元
{$R *.DFM}
procedure TForm1.BtnCalculateClick(Sender:TObject);
label FinalExit, NormalExit, ErrorExit;
var
  dPesticide, dCost, dGain, dArea: array[1..2] of Double;
  dTotal, dLimit, dStatus: Double;
  nError, pLingo, nPointersNow: integer;
  cScript: pchar;
  cMsg: string;
```


**图 13-9　程序界面**

```
  i :integer ;
begin
  //调用 LINGO DLL 求解农药管理问题 ;
  //模型文件 pesticide.lng 在本例中需要与 Delphi 执行文件置于同目录下 ;
  //输入数据从程序的对话框中获得 ;
  for i := 1 to 2 do
  begin
    dPesticide [i] := StrToFloat (TEdit (FindComponent ( pesticide' + inttostr (i))) . Text) ;
    dCost [i]      := StrToFloat (TEdit (FindComponent ( cost' + inttostr (i))) . Text) ;
    dGain [i]      := StrToFloat (TEdit (FindComponent ( gain' + inttostr (i))) . Text) ;
  end ;
  dTotal := StrToFloat (total . Text) ;
  dLimit := StrToFloat (limit . Text) ;

  //创建 LINGO 环境对象
  pLingo := LScreateEnvLng () ;
  if ( pLingo = 0) then
  begin
    ShowMessage ( ' Can' ' t create LINGO environment') ;
    goto FinalExit ;
  end ;

  //打开 LINGO 日志文件
  nError := LSopenLogFileLng ( pLINGO, ' LINGO . log') ;
  if ( nError < > LSERR _ NO _ ERROR _ LNG) then goto ErrorExit ;
```

```
//将指针传递给 LINGO
//@POINTER(1)
nError := LSsetPointerLng(pLINGO, dPesticide[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(2)
nError := LSsetPointerLng(pLINGO, dCost[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(3)
nError := LSsetPointerLng(pLINGO, dGain[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(4)
nError := LSsetPointerLng(pLINGO, dTotal, nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(5)
nError := LSsetPointerLng(pLINGO, dLimit, nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(6)
nError := LSsetPointerLng(pLINGO, dArea[1], nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;
//@POINTER(7)
nError := LSsetPointerLng(pLINGO, dStatus, nPointersNow);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;

//创建 LINGO 命令脚本
cScript := 'SET ECHOIN 1' + Char(10) +
'TAKE pesticide.LNG + Char(10) +
'GO + Char(10) +
'QUIT + Char(10) +
Char(0);

//执行脚本
nError := LSexecuteScriptLng(pLINGO, cScript);
if(nError <> LSERR_NO_ERROR_LNG) then goto ErrorExit;

//关闭日志文件
LScloseLogFileLng(pLINGO);

//出现问题
if((nError <> LSERR_NO_ERROR_LNG) or
(dStatus <> LS_STATUS_GLOBAL_LNG)) then
  ShowMessage('Unable to solve!')
else
```

```
    //无任何问题则在窗口中显示返回结果
    begin
      //返回结果
      for i := 1 to 2 do
      TEdit(FindComponent('area' + inttostr(i))).Text := floatToStr(dArea[i]);
    end;
    goto NormalExit;
    ErrorExit:
      cMsg := Format('LINGO Errorcode:%d', [nError]);
      ShowMessage(cMsg);
      goto FinalExit;
    NormalExit:
      //释放 LINGO 环境对象以避免占用内存
      LSdeleteEnvLng(pLINGO);
    FinalExit:
  end;
```

**点击“计算”按钮后,模型求解并将结果返回到界面上,如图** 13-10 **所示。**


**图** 13-10　**求解后的程序界面**

## 13.15　农药管理问题

### 13.15.1　问题描述

**上述问题是一个典型的线性问题,而本例是与之类似的非线性问题,但是在应用** LINGO **时,用户不会感觉到两者模型求解难度的差异,这也正是** LINGO **软件的优势所在。**

**一个容积为** 100 000 $m^3$ **的湖泊,周围有** 1 000 $hm^2$ **农田,施加在农作物上的农药有一部分流**

失到湖泊中，从而危害到吃鱼的鹰。当地环保部门想知道如何管理农田，才不会对鹰造成危害。生物学家确认湖水中的农药在食物链中被富集。研究结果显示，农药浓度随着食物链呈几何级数增长。设湖水中农药浓度为 C (mg/L)，则湖水中浮游植物 (藻类) 的浓度为 $C^2$，食藻类鱼体内浓度为 $C^3$，鹰体内浓度为 $C^4$。假定 C 不小于 1 mg/L，鹰的最大耐农药浓度为 100 mg/L。

在 1 000 $hm^2$ 的农田上，生长着两种作物，它们具有不同的农药施加量、农药流失率、单位种植面积的收入和费用，具体数据如表 13-11 所示。

表 13-11　数据表

| 作物 | 种植面积 ($hm^2$) | 农药施加量 (kg/$hm^2$) | 农药流失率 (%) | 作物收入 (美元/$hm^2$) | 作物费用 (美元) |
|---|---|---|---|---|---|
| 1 | $X_1$ | 6 | 15 | 300 | $1\,500\sqrt{X_1}$ |
| 2 | $X_2$ | 2.5 | 20 | 150 | $600\sqrt{X_2}$ |

为了确定鹰体内的农药浓度，首先要计算进入湖水中的农药量。种植 1 $hm^2$ 作物 1 需要 6 kg农药，其中 15%流失到湖水中，由作物 1 产生的农药总流失量为 $6\times0.15\times X_1=0.9X_1$。相类似，由作物 2 产生的农药总流失量为 $2.5\times0.20\times X_2=0.5X_2$。湖水的平均农药浓度为 $(0.9X_1+0.5X_2)/200\,000$，换算成 mg/L 为：

$$C=\frac{0.9X_1+0.5X_2}{200}$$

鹰体内的农药浓度是 $C^4$，并要求这个浓度不大于 100 mg/L，则

$$\left(\frac{0.9X_1+0.5X_2}{200}\right)^4\leqslant 100$$

完整的模型为

$$\max\ Z=300X_1-1\,500\sqrt{X_1}+150X_2-600\sqrt{X_2}$$

s.t.

$$\left(\frac{0.9X_1+0.5X_2}{200}\right)^4\leqslant 100$$

$$X_1+X_2\leqslant 1\,000$$

$$X_1\geqslant 0$$

$$X_2\geqslant 0$$

## 13.15.2　LINGO 模型

根据题意，建立如下 LINGO 模型：

```
SETS:
  CROP/1 2/ :PESTICIDE,LOST,COST,GAIN,AREA;
ENDSETS
DATA:
```

```
        PESTICIDE = 6 2.5;
        LOST = 15 20;
        COST = 1500 600;
        GAIN = 300 150;
        TOTAL_AREA = 1000;
        MAX_CONCENTRATION = 100;
        LAKE_VOLUME = 100000;
        RT = 6;
      ENDDATA
      [TOTAL_GAIN] MAX = @SUM(CROP(I):GAIN(I)*AREA(I) - COST(I)*AREA(I)^0.5);
      @SUM(CROP(I):PESTICIDE(I)*AREA(I)*LOST(I)/100)*1000/(LAKE_VOLUME*12
/RT) <= MAX_CONCENTRATION^(1/4);
      @SUM(CROP(I):AREA(I)) <= TOTAL_AREA;
```

## 13.15.3 求解结果

**求解上述模型,得到下列结果：**

```
Local optimal solution found at iteration:          56
Objective value:                              171055.0
```

| Variable | Value | Reduced Cost |
|---|---|---|
| TOTAL_AREA | 1000.000 | 0.000000 |
| MAX_CONCENTRATION | 100.0000 | 0.000000 |
| LAKE_VOLUME | 100000.0 | 0.000000 |
| RT | 6.000000 | 0.000000 |
| PESTICIDE(1) | 6.000000 | 0.000000 |
| PESTICIDE(2) | 2.500000 | 0.000000 |
| LOST(1) | 15.00000 | 0.000000 |
| LOST(2) | 20.00000 | 0.000000 |
| COST(1) | 1500.000 | 0.000000 |
| COST(2) | 600.0000 | 0.000000 |
| GAIN(1) | 300.0000 | 0.000000 |
| GAIN(2) | 150.0000 | 0.000000 |
| AREA(1) | 702.7284 | 0.000000 |
| AREA(2) | 0.000000 | 2683283 |

| Row | Slack or Surplus | Dual Price |
|---|---|---|
| TOTAL_GAIN | 171055.0 | 1.000000 |
| 2 | 0.000000 | 60379.50 |
| 3 | 297.2716 | 0.000000 |

# 参考文献


1 (美) 海思 (Haith D. A.) . 环境系统最优化. 袁铭道, 季民译. 北京: 中国环境科学出版社, 1987

2 (美) 伯拉斯 (N. Buras) . 水资源科学分配. 戴国瑞等译. 北京: 中国水利电力出版社, 1983

3 张超. 水资源系统动态规划. 北京: 中国水利电力出版社, 1986

4 傅国伟. 环境工程手册 (环境规划卷) . 北京: 高等教育出版社, 2003

5 程声通, 孟繁坚, 徐明德. 环境系统分析题解. 北京: 高等教育出版社, 1994

6 洪文, 吴本忠. LINGO 4.0 for Windows 最优化软件及其应用. 北京: 北京大学出版社, 2001