目 录

第-	一章	大学生数学建模竞赛综述	
	第一节	数学建模竞赛的兴起和发展	1
	第二节	数学建模竞赛的组织、培训和论文撰写	3
	第三节	数学模型的建立过程	4
	第四节	几个建模示例	8
	第五节	数学建模竞赛题评析	13
第.	二章	规划与优化模型	20
	第一节	奶制品的生产与销售	20
	第二节	自来水输送与货机装运	23
	第三节	汽车生产与原油采购	27
	第四节	接力队选拔和选课策略	31
	第五节	饮料厂的生产与检修	35
第=	三章	微分方程模型	38
	第一节	根据规律建模	38
	第二节	微元法建模	39
	第三节	模拟近似法建模	39
	第四节	微分方程模型实例	41
第	軍軍	图论模型	49
	第一节	图论的基本概念	49
	第二节	图的存储结构	53
	第三节	树与图的生成树	54
	第四节	网络流及其应用	56
	第五节	图的遍历	63
第三	丘章	数学软件使用初步	66
	第一节	MATLAB	66
	§1		
	§1	_ , _ , , , , , , , , , , , , , , , , ,	
	§1		
	§1	— , , , , , ,	
	§1		
	§1		
	§1		
	§1		
	第二节	•	
	§1	•	
	§1		
	§1		
	§1		
	§1		
	§1	.6 作图	100

§1.7	函数与过程	103
§1.8	流程控制	107
第三节	LINDO 与 LINGO	108
§1.1	LINDO 与 LINGO 简介	108
§1.2	用 LINDO 求解线性规划(LP)问题	109
§1.3	用 LINDO 求解整数规划和二次规划问题	112
§1.4	LINGO 功能简介	114

第一章 大学生数学建模竞赛综述

第一节 数学建模竞赛的兴起和发展

一. 数学建模竞赛的历程

- 20 世纪 60~70 年代进入西方国家的大学(数学建模教材较集中地出现在 70 年代).
- 20 世纪 80 年代初开始进入我国大学; 1987 年出版第 1 本教材(《数学模型》, 姜启源编, 高教社); 80 年代末估计 30~40 所学校开课(数学系, 讲座).
- 1985 年美国大学生数学建模竞赛开始举办, 1989 年我国大学生开始参加这项竞赛.
- 1992 年我国大学生数学建模竞赛开始举办, 2001 年有 26 省(市、自治区)500 多所学校参加.
- 数学建模竞赛与教学相互促进. 数学模型课已作为数学与应用数学、信息与计算科学等专业的基础课

二. 广西及我院参赛情况

- 广西 1994 年开始组织参赛, 当时只有广西大学、桂林电子工业学院参加.
- 2002年广西已有13所本专科院校(1999年设专科组)53个队参赛. 但迄今为止参加美国大学生数模竞赛还是空白.
- 我院 1996 年第一次组队参加全国大学生数学建模竞赛. 获一个三等奖.
- 1998年再次组队(3队)参赛、获广西一等奖一个.
- 1999 年 3 个队,全国二等奖 1 个,广西三等奖 1 个;2000 年 5 个队,全国二等奖 1 个,广西二、三等奖各 1 个;2001 年 6 个队,广西三等奖 2 个;2002 年 6 个队,全国二等奖 1 个,广西一、二、三等奖各 1 个;2003 年 6 个队,全国二等奖 1 个,广西二等奖 2 个、三等奖各 1 个.
- 1. 顺应知识经济时代的潮流、适应高科技发展的需要教育必须反映并满足社会发展的需求
- 计算机技术和数学软件的迅速发展, 为数学建模的应用提供了强有力的工具;
- 数学迅速进入一些诸如经济、生态、人口、交通等领域, 为数学建模开拓了许多新的处女地;
- 计算机技术与数学建模在知识经济中起着如虎添翼的作用;
- "数学是一种关键的,普遍的,可应用的技术" 数学"由研究到工业领域的技术转化,对加强经济竞争具有重要意义";
- "计算和建模重新成为中心课题, 它们是数学科学技术转化的主要途径";
- 教育必须反映并满足社会发展的需求;
- 2. 培养学生用数学的能力, 探索数学教学改革的途径
- 数学教育应该培养学生两种能力: "计算数学"(计算、推导、证明···)和"应用数学"(实际问题建模及模型结果的分析、检验、应用);
- 传统数学教学体系和内容偏重前者, 忽略后者;
- 数学建模引入教学是不打乱现有体系下的教改实验.

数学建模竞赛的迅速发展, 培养了学生创新精神, 推动了高校的教学改革.

三. 数学建模竞赛内容、形式和评奖标准

竞赛内容: 题目由工程技术、管理科学中的实际问题简化而成, 没有事先设定的标准答案, 但留有充分余地供参赛者发挥其聪明才智和创造精神.

竞赛形式: 三名大学生组成一队, 可以自由地收集资料、调查研究, 使用计算机、互联网和任何软件, 在三天时间内分工合作完成一篇论文.

评奖标准: 假设的合理性、建模的创造性、结果的正确性和文字表述的清晰程度.

大学阶段难得的一次近似于"真刀真枪"的训练,模拟了毕业后工作时的情况,既丰富、活跃了 广大同学的课外生活,也为优秀学生脱颖而出创造了条件.

四. 数学建模竞赛培养学生创新精神, 提高学生综合素质

- 运用学过的数学知识和计算机(包括选择合适的数学软件)分析和解决实际问题的能力
- 面对复杂事物的想象力、洞察力、创造力和独立进行研究的能力
- 关心、投身国家经济建设的意识和理论联系实际的学风
- 团结合作精神和进行协调的组织能力
- 勇于参与的竞争意识和不怕困难、奋力攻关的顽强意志
- 查阅文献、收集资料以及撰写科技论文的文字表达能力

竞赛宗旨: 创新意识 团队精神 重在参与 公平竞争

- 答卷按省(市、自治区)和全国两级评奖
- 每年赛题、优秀答卷及获奖名单刊登于次年"数学的实践与认识"和"工程数学"第1期
- 1999年的竞赛命名为"99创维杯全国大学生数学建模竞赛"
- 2000 年的竞赛命名为 "2000 网易杯全国大学生数学建模竞赛"
- 2002 年起竞赛命名为 "2002 高教社杯全国大学生数学建模竞赛"
- 全国组委会网址: http://csiam.edu.cn/mcm/
- 竞赛的社会影响不断扩大

五. 数学建模教学和竞赛的教材、辅导材料及参考书

- 姜启源, 数学模型(第二版), 高等教育出版社, 1993
- 任善强等, 数学模型(第二版), 重庆大学出版社, 1998
- 谭永基等, 数学模型, 复旦大学出版社, 1997
- 刘来福等, 数学模型与数学建模, 北京师范大学出版社, 1997
- 陈义华, 数学模型, 重庆大学出版社, 1995
- 徐全智等, 数学建模入门, 电子科技大学出版社, 1996
- 吴翊等, 数学建模的理论与实践, 国防科技大学出版社, 1999
- 周义仓等, 数学建模实验, 西安交通大出版社, 1999
- 白其峥主编, 数学建模案例分析, 海洋出版社, 1999
- W. F. Lucas 主编, 朱煜民等译, Modules in Applied Mathematics (四卷本, 包括微分方程模型, 政治及有关模型, 离散与系统模型, 生命科学模型), 国防科技大学出版社, 1997(第一卷 1988)
- 姜启源等, 数学实验, 高等教育出版社, 1999
- 李尚志等, 数学实验, 高等教育出版社, 1999
- 乐经良等, 数学实验, 高等教育出版社, 1999
- 谢云逊等, 数学实验, 科学出版社, 1999
- 叶其孝主编, 大学生数学建模竞赛辅导教材, 湖南教育出版社, 1993
- 叶其孝主编, 数学建模教育与国际数学建模竞赛, 《工科数学》杂志社, 1994
- 叶其孝主编、大学生数学建模竞赛辅导教材(二)、湖南教育出版社、1997
- 叶其孝主编, 大学生数学建模竞赛辅导教材(三), 湖南教育出版社, 1998
- 李大潜主编, 中国大学生数学建模竞赛, 高等教育出版社, 1998
- 李尚志主编, 数学建模竞赛教程, 江苏教育出版社, 1996

六. 数学建模相关期刊

- 数学的实践与认识(中, 季刊)
- 工程数学(季刊)

- Mathematical and Computer Modelling (美, 半月刊)
- Applied Mathematical Modelling (美, 月刊)
- The Journal of Undergraduate Mathematics and its Applications(UMAP, 美, 季刊)
- Mathematical Models and Methods in Applied Science(新加坡, 双月刊)

七. 全国数学建模教学与应用会议

- 第1届(1986, 西安);
- 第2届(1988, 衡阳);
- 第 3 届(1991, 长沙);
- 第4届(1993,太原);
- 第5届(1996, 长春);
- 第6届(1998, 南昌);
- 第7届(2000, 郑州);
- 第8,9届(8月,北京);
- 第10届(2003,大连)

八. International Conference on the Teaching of Mathematical Modeling and Applications (ICTMA)

ICTMA1(1983)每两年一次,欧洲;

ICTMA8(1997, 澳);

ICTMA9(1999, 葡);

ICTMA10(2001, 中国北京)

九. 数学建模竞赛中常用的模型

- 初等数学方法建模(代数、几何、初等概率方法)
- 量纲分析法建模
- 微分法建模(静态优化模型)
- 微分方程模型(动态模型, 常微部分)
- 差分方程模型
- 层次分析法建模
- 随机模型(概率分布方法建模)
- 微分方程模型(偏微部分)
- 稳态模型(稳定性方法建模)
- 图的方法建模(简单的图论方法的应用)
- 逻辑方法建模(合作对策模型等)
- 马氏链模型
- 随机服务模型
- 数学规划模型
- 回归模型

第二节 数学建模竞赛的组织、培训和论文撰写

一. 数学建模竞赛的培训内容

1)建模的基本概念和方法(数学建模课程的主要内容)

2)建模过程中常用的数学方法(微积分、代数、概率外),主要有: 计算方法(如数值微分和积分、微分方程数值解、代数方程组解法),优化方法(如线性、非线性规划),数理统计(如假设检验、回归

分析), 图论(如最短路)等.

只要求知道实际问题与这些数学知识之间的对应关系(如哪些问题可用线性规划求解,或线性规划可解决哪些问题),以及用它们建立模型的方法,基本上不必涉及模型的求解.

- 3)合适的数学软件的基本用法. 基本上能完成上述方法的软件, 如 Mathematica, Maple 等.
- 4) 历届赛题的研讨.
- 5)撰写数学建模论文的练习.

其中 1)2)以教师讲授为主, 3)5)以学生实习为主, 4)以学生讨论、教师辅导为主.

二. 数学建模竞赛组队的方式

- 尽可能地让不同专业的学生组成一队, 以利学科交叉;
- 尽可能地让能力、素质方面不同的学生(创新能力强的, 认真踏实的, 有组织能力的, 文笔好的, …)组成一队, 以利优势互补;
 - 尽可能地让学生在队内充分磨合, 达成默契, 形成"领袖".

三. 数学建模竞赛期间的注意事项

- 吃透题意, 确定题目;
- 查阅资料、实际调查要适度;
- 把握好用现成的模型和方法,与自己创新的模型和方法之间的关系;
- 保证基本模型和求解的完成, 在此基础上完善或改进;
- 根据建模的要求, 可以增加、删除甚至修改题目的条件;
- 论文主体由一人完成, 并早些开始写作.

四. 写好论文(答卷)的注意事项

- 完整——摘要;问题提出(用自己的语言);问题分析;模型假设;模型建立;模型求解(算法设计和计算机实现);结果(数据、图形);结果分析和检验(如误差分析、统计检验、灵敏性检验);优缺点,改进方向等,附录(程序、更多的计算结果、复杂的推导、证明等);
 - 摘要——主要模型(名称)、方法和结果、解决了什么问题、有何特色等:
 - 表述清晰、简明, 给出数学符号的确切含义、模型假设的理由等.

第三节 数学模型的建立过程

一. 什么是数学模型?

我们常见的模型

玩具、照片… —— 实物模型

风洞中的飞机… —— 物理模型

地图、电路图… — 符号模型

模型是为了一定目的,对客观事物的一部分进行简缩、抽象、提炼出来的原型的替代物.模型集中反映了原型中人们需要的那一部分特征.

二. 你碰到过的数学模型——"航行问题"

甲乙两地相距 750 公里, 船从甲到乙顺水航行需 30 小时, 从乙到甲逆水航行需 50 小时, 问船的速度是多少.

用x表示船速,y表示水速,列出方程:

 $(x + y) \times 30 = 750$

 $(x-y) \times 50 = 750$

求解得到 x=20, y=5, 答: 船速每小时 20 公里.

三. 航行问题建立数学模型的基本步骤

- 作出简化假设(船速、水速为常数);
- 用符号表示有关量(x, y 表示船速和水速);
- 用物理定律(匀速运动的距离等于速度乘以时间)列出数学式子(二元一次方程);
- 求解得到数学解答(x=20, v=5);
- 回答原问题(船速每小时 20 公里).

四. 数学模型(Mathematical Model)和数学建模 Mathematical Modeling)

数学模型:对于一个现实对象,为了一个特定目的,根据其内在规律,作出必要的简化假设,运用适当的数学工具,得到的一个数学结构.

数学建模:建立数学模型的全过程(包括建立、求解、分析、检验).

五. 数学建模的重要意义

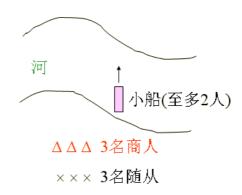
- 电子计算机的出现及飞速发展.
- 数学以空前的广度和深度向一切领域渗透.

数学建模作为用数学方法解决实际问题的第一步, 越来越受到人们的重视.

六. 建模示例

例1 商人们怎样安全过河

问题(智力游戏)



随从们密约, 在河的任一岸, 一旦随从的人数比商人多, 就杀人越货. 但是乘船渡河的方案由商人决定.商人们怎样才能安全过河?

问题分析 多步决策过程

决策——每一步(此岸到彼岸或彼岸到此岸)船上的人员

要求——在安全的前提下(两岸的随从数不比商人多), 经有限步使全体人员过河

模型构成

x₁~第 k 次渡河前此岸的商人数

 v_k ~第 k 次渡河前此岸的随从数

 $x_k, y_k=0,1,2,3; k=1,2,...$

 $s_k=(x_k,y_k)$ 是过程的状态

S 是允许状态集合, $S=\{(x,y)|x=0,y=0,1,2,3;x=3,y=0,1,2,3;x=y=1,2\}$

 u_k 是第 k次渡船上的商人数

 v_k 是第 k 次渡船上的随从数

 $u_k, v_k=0,1,2; k=1,2,...$

 d_k =(u_k , v_k) 是决策, D={(u, v)| u+v=1, 2} 是允许决策集合

 $s_{k+1} = s_k + (-1)^k d_k$ 是状态转移律

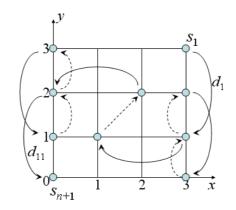
多步决策问题

求 $d_k \in (k=1,2,n)$, 使 $s_k \in S$ 按转移律, 由 $s_1=(3,3)$ 到达 $s_{n+1}=(0,0)$.

模型求解

- 1. 穷举法. 编程上机.
- 2. 图解法.

状态 $s=(x,y)\sim 16$ 个格点, 允许状态 $S\sim 10$ 个圆点, 允许决策 $D\sim 8$ 动 1 或 2 格, k 奇,左下移; k 偶,右上移, $d_1,...d_{11}$ 给出安全渡河方案.



★考虑 4 名商人各带一随从的情况.

例 2 如何预报人口的增长

世界人口增长概况

年	1625	1830	1930	1960	1974	1987	1999
人口(亿)	5	10	20	30	40	50	60

中国人口增长概况

年	1908	1933	1953	1964	1982	1990	1995
人口(亿)	3	4.7	6	7	10.1	11.3	12

研究人口变化规律, 控制人口过快增长.

指数增长模型

1. 常用的计算公式

今年人口 x_0 , 年增长率 r,k年后人口

$$x_k = x_0 (1+r)^k$$

2. 马尔萨斯(1788--1834)提出的指数增长模型(1798)

x(t)是 t 时刻的人口, r 是人口(相对)增长率(常数). 由 $x(t + \Delta t) - x(t) = rx(t)\Delta t$ 解得

$$x(t) = x_0 e^{rt}.$$

设
$$\frac{dx}{dt} = rx$$
, $x(0) = x_0$, 得

$$x(t) = x_0(e^r)^t \approx x_0(1+r)^t$$
.

随着时间增加人口按指数规律无限增长.

指数增长模型的应用及局限性

- 与 19 世纪以前欧洲一些地区人口统计数据吻合
- 适用于 19 世纪后迁往加拿大的欧洲移民后代

- 可用于短期人口增长预测
- 不符合 19 世纪后多数地区人口增长规律
- 不能预测较长期的人口增长过程 19 世纪后人口数据是人口增长率 r 不是常数(逐渐下降)

3. 阻滯增长模型 (Logistic 模型)

人口增长到一定数量后,增长率下降的原因:资源、环境等因素对人口增长的阻滞作用且阻滞作用随人口数量增加而变大.

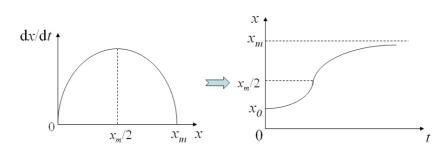
假定: r(x)=r-sx(r,s>0), r 是固有增长率(x 很小时), x_m 是人口容量(资源、环境能容纳的最大数量,

所以
$$r(x_m) = 0$$
, 进而 $s = \frac{r}{x_m}$, 所以有 $r(x) = r(1 - \frac{x}{x_m})$. 因为 $\frac{dx}{dt} = rx$, 所以

$$\frac{dx}{dt} = r(x)x = rx(1 - \frac{x}{x_{m}}),$$

解得

$$x(t) = \frac{x_m}{1 + (\frac{x_m}{x_0} - 1)e^{-rt}}$$



 $x(t)\sim S$ 形曲线, x 增加先快后慢.

模型的参数估计

用指数增长模型或阻滞增长模型作人口预报、必须先估计模型参数 r 或 r, x_m

• 利用统计数据用最小二乘法作拟合

例: 美国人口数据(单位~百万)

1790 1800 1810 1820 1830 1950 1960 1970 1980 3.9 5.3 7.2 9.6 12.9 150.7 179.3 204.0 226.5 r=0.2072, x_m =464

• 专家估计

模型检验

用模型预报 1990 年美国人口, 与实际数据比较.

由 $x(1990) = x(1980) + \Delta x = x(1980) + rx(1980)[1 - x(1980) / x_m]$ 得 x(1990) = 250.5,实际为 251.4 (百万).

模型应用——人口预报

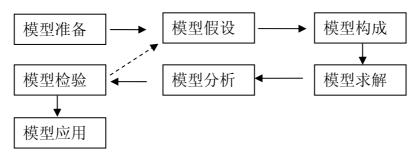
用美国 1790~1990 年人口数据重新估计参数, r=0.2083, x_m =457.6, 由此得到 x(2000)=275.0, x(2010)=297.9

数学建模的方法和步骤

基本方法:

- •机理分析 根据对客观事物特性的认识、 找出反映内部机理的数量规律.
- •测试分析 将研究对象看作"黑箱",通过对量测数据 的统计分析, 找出与数据拟合最好的模型 机理.
 - •二者结合 机理分析建立模型结构,测试分析确定模型参数.

分析没有统一的方法, 主要通过实例研究(Case Studies)来学习. 以下建模主要指机理分析数学建模的一般步骤:



七. 怎样学习数学建模

数学建模与其说是一门技术, 不如说是一门艺术

技术大致有章可循

艺术无法归纳成普遍适用的准则

数学建模需要有丰富的想象力、敏锐的洞察力、准确的判断力和大胆的创新意识.

开始学习建模时需要学习、分析、评价、改进别人作过的模型, 然后亲自动手, 认真作几个实际题目.

第四节 几个建模示例

例 1. 录像机计数器的用途

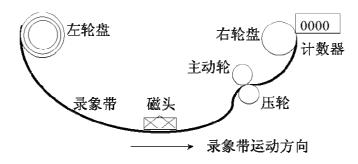
问题: 经试验,一盘录像带从头走到尾,时间用了 183 分 30 秒,计数器读数从 0000 变到 6152. 在一次使用中录像带已转过大半,计数器读数为 4580,问剩下的一段还能否录下 1 小时的节目?

要求: 不仅回答问题, 而且建立计数器读数与 录像带转过时间的关系.

思考: 计数器读数是均匀增长的吗?

问题分析

录像机计数器的工作原理



录像带运动→右轮盘半径增大→计数器读数增长变慢→录像带运动速度是常数→右轮转速不是常数.

观察得出: 计数器读数增长越来越慢!

模型假设

- 录像带的运动速度是常数 v;
- 计数器读数 n 与右轮转数 m 成正比, 记 $m=k_n$;
- 录像带厚度(加两圈间空隙)为常数 w;
- 空右轮盘半径记作 r;
- 时间 *t*=0 时读数 *n*=0.

建模目的

建立时间 t 与读数 n 之间的关系(设 V, k, w, r 为已知参数)

模型建立

建立t与n的函数关系有多种方法

1. 右轮盘转第 i 圈的半径为 r+wi, m 圈的总长度 等于录像带在时间 t 内移动的长度 vt, 所以

$$\sum_{n=1}^{\infty} 2\pi (r + wi) = vt, m = nk$$

由此得

$$t = \frac{\pi w k^2}{v} n^2 + \frac{2\pi r k}{v} n$$

2. 考察右轮盘面积的变化、等于录像带厚度乘以转过的长度、即

$$\pi[(r+wkn)^2-r^2]=wvt$$

由此得

$$t = \frac{\pi w k^2}{v} n^2 + \frac{2\pi r k}{v} n$$

3. 考察 t 到 t+dt 录像带在右轮盘缠绕的长度, 有

$$(r + wkn)2\pi kdn = vdt$$

由此得

$$t = \frac{\pi w k^2}{v} n^2 + \frac{2\pi r k}{v} n$$

思考

- 1. 三种建模方法得到同一结果, 但仔细推算会发现稍有差别, 请解释.
- 2. 模型中有待定参数 r, w, v, k, 确定参数的一种办法是测量或调查, 试设计测量方法.

参数估计

确定参数的另一种方法——测试分析

将模型改记作 $t = an^2 + bn$.

只需估计a, b 理论上,已知t=183.5, n=6152,再有一组(t, n)数据即可;实际上,由于测试有误差,最好用足够多的数据作拟合.

现有一批测试数据:

t	0	20	40	60	80	100	120	140	160	183.5
n	0000	1153	2045	2800	3466	4068	4621	5135	5619	6152

用最小二乘法可得 $a=2.51\times10^{-6}$, $b=1.44\times10^{-2}$.

模型检验

应该另外测试一批数据检验模型:

$$t = an^2 + bn$$
, $(a=2.51 \times 10^{-6}, b=1.44 \times 10^{-2})$

模型应用

- 1. 回答提出的问题: 由模型算得 n = 4580 时 t = 118.5 分, 剩下的录像带能录 183.5-118.5 = 65 分钟的节目.
- 2. 揭示了"t 与 n 之间呈二次函数关系"这一普遍规律,当录像带的状态改变时,只需重新估计 a, b 即可.

例 2. 市场经济中的蛛网模型

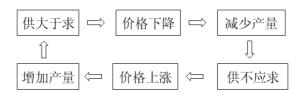
在自由贸易市场上你注意过这样的现象吗:一个时期以来某种消费品如猪肉的上市量远大于需求,由于销售不畅导致价格下降,生产者发现养猪赔钱,于是转而经营其它农副业.过一段时间猪肉上市量就会大减,供不应求将导致价格上涨.生产者看到有利可图,又重操旧业,这样下一个时期会重现供大于求、价格下降的局面.在没有外界干预的情况下,这种现象将如此循环下去.

在完全自由竞争的市场经济中上述现象通常是不可避免的. 因为商品的价格呈由消费者的需求关系决定的,商品数量越多价格越低. 而下一时期商品的数量由生产者的供应关系决定,商品价格越低生产的数量就越少. 这样的需求和供应关系决定了市场经济中商品的价格和数量必然是振荡的. 在现实世界里这样的振荡会出现不同的形式,有的振幅渐小趋向平稳,有的则振幅越来越大,如果没有外界如政府的干预,将导致经济崩溃.

请你对上述现象进行分析,给出市场经济趋于稳定的条件.并建立数学模型,对结果进行解释,并讨论当市场经济不稳定时政府可以采取什么样的干预措施.并对你的模型作适当推广.

问题分析

现象



问题

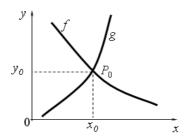
- ◆ 描述商品数量与价格的变化规律.
- ◆ 商品数量与价格的振荡在什么条件下趋向稳定.
- ◆ 当不稳定时政府能采取什么干预手段使之稳定.

模型建立

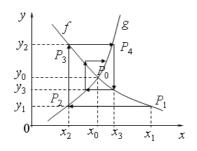
 x_k 一第 k 时段商品数量; y_k 一第 k 时段商品价格 消费者的需求关系→需求函数 y_k = $f(x_k)$ 减函数.

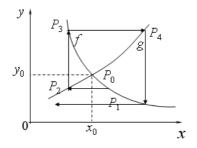
生产者的供应关系→供应函数 $x_{k+1}=h(y_k)$, $y_k=g(x_{k+1})$ 增函数.

f与g的交点 $P_0(x_0,y_0)$ ——平衡点,一旦 $x_k=x_0$,则 $y_k=y_0, x_{k+1}, x_{k+2}, \dots = x_0, y_{k+1}, y_{k+2}, \dots = y_0$.



设 x_1 偏离 $x_0, x_1 \rightarrow y_1 \rightarrow x_2 \rightarrow y_2 \rightarrow x_3 \rightarrow ..., x_k \rightarrow x_0, y_k \rightarrow y_0, P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow ... \rightarrow P_0$.





 $K_f < K_g, P_0$ 是稳定平衡点

 $K_{P}K_{g}, P_{0}$ 是不稳定平衡点

在 Po 点附近用直线近似曲线

$$y_k = f(x_k) \rightarrow y_k - y_0 = -\alpha(x_k - x_0) \ (\alpha > 0)$$

$$x_{k+1} = h(y_k) \rightarrow x_{k+1} - x_0 = \beta(y_k - y_0) \ (\beta > 0)$$

曲
$$x_{k+1} - x_0 = -\alpha \beta (x_k - x_0)$$
 得 $x_{k+1} - x_0 = (-\alpha \beta)^k (x_1 - x_0)$

当
$$\alpha\beta$$
<1时, α (= K_f)< $\frac{1}{\beta}$ (= K_g), $x_k \to x_0$, P_0 稳定

当
$$\alpha\beta > 1$$
时, $\alpha(=K_f) > \frac{1}{\beta}(=K_g)$, $x_k \to \infty$, P_0 不稳定

结果解释

考察 α , β 的含义

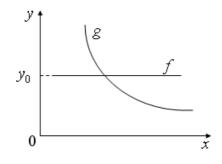
$$y_k - y_0 = -\alpha(x_k - x_0)$$
, α ——商品数量减少 1 单位,价格上涨幅度

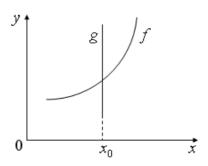
$$x_{k+1} - x_0 = \beta(y_k - y_0), \beta$$
——价格上涨 1 单位, (下时段)供应的增量

 α ——消费者对需求的敏感程度, α 小, 有利于经济稳定 β ——生产者对价格的敏感程度, β 小, 有利于经济稳定 由此 $\alpha\beta$ <1, 经济稳定

经济不稳定时政府的干预办法

- 1. ϕ α 尽量小, 如 α =0, 需求曲线变为水平, 以行政手段控制价格不变
- 2. 使 β 尽量小,如 β =0,供应曲线变为竖直,靠经济实力控制数量不变





例 3. 报童模型

报童每天清晨从报站批发报纸零售,晚上将没有卖完的报纸退回.设每份报纸的批发价为 b,零售价为 a,退回价为 c,且设 a > b > c > 0.因此,报童每出售一份报纸赚钱(a - b),退回一份报纸赔(b - c),报童每天如果批发太少,不够卖的话就会少赚钱;如果批发的报纸太多,卖不完的话就会赔钱.报童

应如何确定他每天批发的报纸的数量,才能获得最大的收益?

问题分析

购进太多→卖不完退回→赔钱

购进太少→不够销售→赚钱少

应根据需求确定购进量,是否存在一个合适的购进量?

每天需求量是随机的, 所以每天收入是随机的.

优化问题的目标函数应是长期的日平均收入,等于每天收入的期望.

调查需求量的随机规律——每天需求量为 r 的概率 f(r). r=0.1,2...

模型建立

- 设每天购进 n 份, 日平均收入为 G(n).
- 已知售出一份赚 a-b; 退回一份赔 b-c.

 $r \leq n$ 一售出 r, 退回 n-r, 赚(a-b)r, 赔(b-c)(n-r)

r>n→售出 *n*, 赚(*a-b*)*n*

$$G(n) = \sum_{r=0}^{n} [(a-b)r - (b-c)(n-r)]f(r) + \sum_{r=n+1}^{\infty} (a-b)nf(r)$$

求 n 使 G(n)最大.

模型求解

将r视为连续变量,f(r)=p(r)(概率密度),

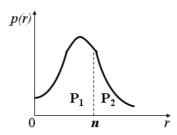
$$G(n) = \int_0^n [(a-b)r - (b-c)(n-r)]p(r)dr + \int_0^\infty (a-b)np(r)dr.$$

由上可得
$$\frac{dG}{dn}$$
 = $(a-b)np(n) - \int_0^n (b-c)p(r)dr - (a-b)np(n) + \int_n^\infty (a-b)p(r)dr$

$$=-(b-c)\int_0^n p(r)dr+(a-b)\int_r^\infty p(r)dr.$$

$$\Rightarrow \frac{dG}{dn} = 0$$
 得

$$\frac{\int_0^n p(r)dr}{\int_n^\infty p(r)dr} = \frac{a-b}{b-c}.$$



结果解释

$$\int_{0}^{n} p(r)dr = P_{1}, \int_{n}^{\infty} p(r)dr = P_{2}, \ \mathbb{R} \ n \ \text{th}$$

$$\frac{P_1}{P_2} = \frac{a-b}{b-c}$$

其中 a-b~售出一份赚的钱, b-c~退回一份赔的钱.

$$(a-b) \uparrow \Rightarrow n \uparrow, (b-c) \uparrow \Rightarrow n \downarrow$$

第五节 数学建模竞赛题评析

例 1 投资的收益和风险(1998 年全国大学生数学建模竞赛题目 A 题)

市场上有 n 种资产(如股票、债券、…) S_i (i=1, …, n) 供投资者选择,某公司有数额为 M 的一笔相当大的资金可用作一个时期的投资。公司财务分析人员对这 n 种资产进行了评估,估算出在这一时期内购买 S_i 的平均收益率为 r_i ,并预测出购买 S_i 的风险损失为 q_i 。考虑到投资越分散,总的风险越小,公司确定,当用这笔资金购买若干种资产时,总体风险可用所投资的 S_i 中最大的一个风险来度量

购买 S_i 要付交易费,费率为 p_i ,并且当购买额不超过给定值 u_i 时,交易费按购买 u_i 计算(不买当然无须付费). 另外,假定同期银行存款利率是 r_0 ,且既无交易费又无风险. (r_0 =5%)

1) 以知 *n*=4 时的相关数据如下:

S_i	$r_i(\%)$	q_i (%)	p_i (%)	$u_i(元)$
S_1	28	2.5	1	103
S_2	21	1.5	2	198
S_3	23	5.5	4.5	52
S_4	23	2.6	6.5	40

试给该公司设计一种投资组合方案,即用给定的资金 M,有选择地购买若干种资产或存银行生息,使净收益尽可能大,而总体风险尽可能小.

2) 试就一般情况对以上问题进行讨论,并利用以下数据进行计算.

S_i	$r_i(\%)$	q_i (%)	$p_i(\%)$	$u_i(元)$
S_1	9.6	42	2.1	181
S_2	18.5	54	3.2	407
S_3	49.4	60	6.0	428
S_4	23.9	42	1.5	549
S_5	8.1	1.2	7.6	270
S_6	14	39	3.4	397
S_7	40.7	68	5.6	178
S_8	31.2	33.4	3.1	220
S_9	33.6	53.3	2.7	475
S_{10}	36.8	40	2.9	248
S_{11}	11.8	31	5.1	195
S_{12}	9	5.5	5.7	320
S_{13}	35	46	2.7	267
S_{14}	9.4	5.3	4.5	328
S_{15}	15	23	7.6	131

问题分析

这是一个优化问题,可设决策变量为每种资产的投资额(投资组合),目标是使得净收益最大,且整体风险最小.但是二者矛盾.可采取下面几种方案考虑:

- 在一定风险下收益最大的决策.
- 在一定收益下风险最小的决策.
- 收益和风险按一定比例组合最优的决策.

这时可得到多组解,由决策者进行选择.冒险型投资者从中选择高风险下收益最大的决策,而保守型投资者则可从低风险下的决策中选取.

模型建立

用数学符号和式子表述决策变量、构造目标函数、确定约束条件.

决策变量: x_i 一对 $S_i(i=0,1,\cdots n)$ 的投资, S_0 表示存入银行.

目标函数:

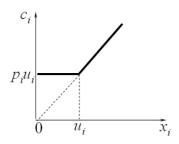
- · 总收益—投资 S_i的净收益减去交易费, 对 i 求和.
- 总体风险—投资 S_i 的风险, 对 i 求最大值.

约束条件: 对 S_i 的投资 x_i 加交易费, 对i求和, 不超过给定资金M.

1) 投资 S_i 的交易费、净收益、风险、资金表达式 交易费

$$c_{i}(x_{i}) = \begin{cases} 0, & x_{i} = 0 \\ p_{i}u_{i}, & 0 < x_{i} < u_{i} \\ p_{i}x_{i}, & x_{i} \ge u_{i} \end{cases}$$

图形如下:



净收益(收益率 r_i)

$$R_i(x_i) = r_i x_i - c_i(x_i)$$

风险(风险损失率 q_i)

$$Q_i(x_i) = q_i x_i$$

资金

$$f_i(x_i) = x_i + c_i(x_i), i = 0,1,...,n, c_0(x_0) = q_0 = 0$$

2) 投资方案总收益、总体风险、资金表达式 投资方案

$$x = (x_0, x_1, \cdots x_n)$$

总收益

$$R(x) = \sum_{i=0}^{n} R_i(x_i)$$

总体风险

$$Q(x) = \max_{0 \le i \le n} Q_i(x_i)$$

资金

$$F(x) = \sum_{i=0}^{n} f_i(x_i)$$

3)两目标(总收益、总体风险)优化模型

$$\min_{x} \left\{ \begin{pmatrix} Q(x) \\ -R(x) \end{pmatrix} \middle| F(x) = M, x \ge 0 \right\}$$

4)单目标优化模型

模型 M1: 确定风险水平 \overline{q} , 记 $k = \overline{q}M$,

 $\max R(x)$

s.t. $Q(x) \leq k$

 $F(x)=M, x \ge 0$

模型 M2: 确定盈利水平 \bar{r} , 记 $h=\bar{r}M$,

 $\min Q(x)$

s.t. $R(x) \ge h$

 $F(x)=M, x \ge 0$

模型 M3: 确定投资者对风险—收益的相对偏好参数 ρ>0,

 $\min \quad S(x) = \rho Q(x) - (1 - \rho)R(x)$

s.t. $F(x)=M, x \ge 0$

模型简化

交易费 $c_i(x_i)$ 当 u_i 很小且 M 很大时,有

 $c_i(x_i)=p_ix_i$.

对资金约束,由

$$F(x) = \sum_{i=0}^{n} f_i(x_i), \quad f_i(x_i) = x_i + c_i(x_i), \quad F(x) = M$$

得到

$$\sum_{i=0}^{n} (1+p_i)x_i = M$$

设M=1,投资 S_i 的比例

$$y_i = (1 + p_i)x_i, i = 0,1,...,n$$

由此,模型 M1 简化为

$$\max \sum_{i=0}^{n} (r_i - p_i) x_i$$

s.t.
$$q_i x_i \le k, i = 1, 2, ..., n$$

$$\sum_{i=0}^{n} (1 + p_i) x_i = 1, \ x \ge 0$$

这是一个线性规划模型(LP1).

模型 M2 简化为

$$\min \quad \max_{0 \le i \le n} (q_i x_i)$$

s.t.
$$\sum_{i=0}^{n} (r_i - p_i) x_i \ge h$$

$$\sum_{i=0}^{n} (1 + p_i) x_i = 1, x \ge 0$$

这是一个极大极小规划模型. 引入人工变量 x_{n+1} , 上面模型可化为线性规划模型(LP2):

min x_{n+1}

s.t.
$$q_i x_i \leq x_{n+1}, i=1,...,n$$

$$\sum_{i=0}^n (r_i - p_i) x_i \geq h$$

$$\sum_{i=0}^{n} (1 + p_i) x_i = 1, x \ge 0$$

模型 M3 的简化为

min
$$L(x) = \rho x_{n+1} - (1-p) \sum_{i=0}^{n} (r_i - p_i) x_i$$

s.t.
$$\sum_{i=0}^{n} (1 + p_i) x_i = 1, x \ge 0$$

 $q_i x_i \leq x_{n+1}, i=1,...,n$

这是一个线性规划模型(LP3).

2.5

26.7327

0

模型求解

LP1, LP2, LP3 都很容易用 MATLAB, MATHEMATICA 或其它数学软件求解.

风险 K(%) 收益 R(%) y_0 y_1 y_2 *y*₃ *y*₄ 0 5.0000 1.0000 0 0 0 0 0.1 7.5528 0.8316 0.04040.0680 0.0190 0.0410 0.2 10.1055 0.6633 0.0808 0.1360 0.0380 0.0819 0.3 12.6583 0.4949 0.1212 0.2040 0.0570 0.1229 0.4 15.2110 0.3266 0.1616 0.2720 0.0760 0.1638 0.5 17.7638 0.1582 0.2020 0.3400 0.0950 0.2048 0.6 20.1908 0 0.2424 0.4080 0.2356 0.11400.7 20.6607 0 0.2828 0.4760 0.1330 0.1082 0 0.3232 0.5440 0.8 21.1243 0.1328 0 0.9 0 0.0244 21.5520 0.3636 0.6120 0 1.0 21.9020 0.4040 0.5960 0 1.5 23.5392 0 0.6060 0.3940 0 0 2.0 0 0 0 25.1765 0.8080 0.1920 2.1 25.5039 0 0.8484 0.1516 0 0 2.2 25.8314 0 0.8888 0.1112 0 0 2.3 26.1588 0 0.9292 0.0708 0 0 26.4863 0.9696 0.0304 0 2.4 0 0

LP1 求解结果: 风险水平取 $k=0\sim2.5\%$, 得投资比例 $v_0\sim v_4$.

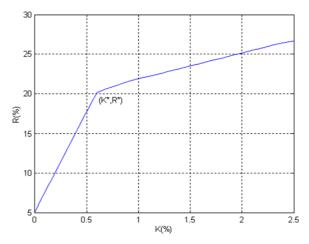
LP1 风险 K 和收益 R 的关系

1.0000

0

0

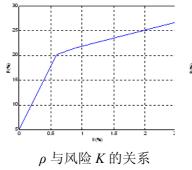
0



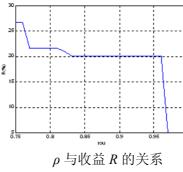
对于风险和收益没有特殊偏好的投资者来说, 应该选择图中曲线的拐点(K^* , R^*), 大约是 K*=0.6%, R*=20%.

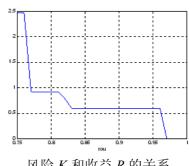
风险 K(%) 收益 R(%) y_0 y_1 *y*₃ *y*₄ y_2 36.7327 1.0000 0 0.76 2.1752 0 0 0 0.77 0.9225 21.6482 0 0.3727 0.6273 0 0 0.81 0.9225 0.3727 0.6273 21.6482 0 0 0 21.0599 0.5338 0.1494 0.82 0.7849 0 0.3171 0 0.83 0.5940 20.1624 0 0.2400 0.4039 0.1129 0.2432 0.96 0.5940 20.1624 0 0.2400 0.4039 0.1129 0.2432 0.97 5.0000 1.0000 0 0 0.0000 0 0

LP3 求解结果: 偏好系数 ρ 取 0.76 \sim 0.97.



结果与 LP1 相同.





风险 K 和收益 R 的关系

例 2 节水洗衣机 (1996 年全国大学生数学建模竞赛题目 B 题)

我国淡水资源有限,节约用水人人有责。洗衣在家庭用水中占有相当大的份额,目前洗衣机已 非常普及, 节约洗衣机用水十分重要. 假设在放人衣物和洗涤剂后洗衣机的运行过程为: 加水一漂 洗一脱水—加水—漂洗—脱水—…—加水—漂洗—脱水(称"加水—漂洗—脱水"为运行一轮). 请 为洗衣机设计一种程序(包括运行多少轮、每轮加水量等),使得在满足一定洗涤效果的条件下)总用 水量最少. 选用合理的数据进行计算. 对照目前常用的洗衣机的运行情况, 对你的模型和结果作出 评价.

问题分析

- 不要求对洗衣的微观机制(物理、化学方面)深入研究, 只需从宏观层次去把握.
- 洗衣的基本原理(宏观上)是用洗涤剂通过漂洗把吸附在衣物上的污物溶于水中、再脱去污水

带走污物.

- 洗衣的过程是通过"加水——漂洗——脱水"程序的反复运行, 使残留在衣物上的污物越来越少, 直到满意的程度.
 - 洗涤剂也是不希望留在衣物上的东西, 可将"污物"定义为衣物上原有污物与洗涤剂的总和.

基本假设

- 每轮漂洗后污物均匀地溶于水中;
- 每轮脱水后衣物含水量为常数 c;

基本模型

x₀——初始污物量

 u_k ——第 k 轮加水量;

 x_k ——第 k 轮脱水后污物量(k=1,2,...)

每轮脱水前后污物在水中的浓度不变

$$\frac{x_0}{u_1} = \frac{x_1}{c}, \quad \frac{x_1}{u_2 + c} = \frac{x_2}{c}, \dots, \quad \frac{x_{n-1}}{u_{n-1} + c} = \frac{x_n}{c}$$

由此得

$$\frac{x_n}{x_0} = \frac{c^n}{u_1(u_2 + c)\cdots(u_n + c)}$$

在最终污物量与初始污物量之比 x_n/x_0 小于给定的 ε (清洁度)条件下, 求各轮加水量 u_k (k=1,···n),使总用水量最小. 由此得到模型

$$\min_{u_k} \sum_{k=1}^n u_k$$

s.t.
$$\frac{c^n}{u_1(u_2+c)\cdots(u_n+c)} < \varepsilon.$$

即

$$\min_{u_k} u_1 + (u_2 + c) + \dots + (u_n + c)$$

s.t.
$$u_1(u_2 + c) \cdots (u_n + c) = a$$
.

由于几何平均值小于 (等于)算术平均值,得 $u_1 = u_2 + c = \cdots = u_n + c$. 所以第 $2\sim n$ 轮加水量 $u_k = u$ (常数),第 1 轮加水量 $u_1 = u + c$. 模型简化为

 $\min nu + c$

s.t.
$$\left(\frac{c}{u+c}\right)^n < \varepsilon$$
.

又 u=cx, 进一步有

 $\min_{n,u} nx$

s.t.
$$\left(\frac{1}{1+x}\right)^n < \varepsilon$$
.

而

$$n = \frac{\alpha}{\ln(1+x)}$$
, $\alpha = -\ln \varepsilon$.

得

$$\min_{x} \frac{\alpha x}{\ln(1+x)}.$$

所以 $x \rightarrow 0$. $u=cx \rightarrow 0$, $n \rightarrow \infty$.

考虑加水量限制 $v_1 \le u \le v_2$,得 $n_{\min} \le n \le n_{\max}$. 其中 $n_{\min} = [\frac{\alpha}{\ln(1+v_2/c)}]+1$,

$$n_{\text{max}} = \left[\frac{\alpha}{\ln(1 + v_1/c)}\right] + 1.$$

作进一步的考虑. m 是衣物重量,每轮脱水后衣物含水量 c=am,浸泡衣物所需水量 d=bm,洗衣机最小注入水量 v_0 ,洗衣机水量下限 $v_1=v_0+bm$.

用以下实验数据测定 a, b.

m(kg)	0.5	1.0	1.5	2.0	2.5	3.0	3.5
c(l)	0.30	0.61	0.92	1.20	1.42	1.88	2.15
d(l)	2.56	5.02	7.48	9.87	12.3	15.7	18.6

得到 a=0.6, b=5.0.

模型结果(单位: 衣物重量(kg); 用水量 (l))

衣物重量	洗衣次数	第1次用水量	第2次用水量	第3次用水量	总用水量	清洁度 ε
1.0	2	29.0	27.4		56.4	0.003
1.2	2	30.1	28.1		58.2	0.004
1.4	2	31.7	29.4		61.1	0.005
1.6	2	36.2	33.6		69.8	0.005
1.8	2	40.7	37.9		78.6	0.005
2.0	3	34.3	31.1	31.1	96.5	0.0008
2.2	3	35.4	31.8	31.8	99.0	0.0009
2.4	3	36.4	32.6	32.6	101.6	0.0011
2.6	3	37.5	33.3	33.3	104.1	0.0013
2.8	3	38.5	34.1	34.1	106.7	0.0015
3.0	3	39.6	34.8	34.8	109.2	0.0017

与实际比较:海棠洗衣机 XQB42-1型,3kg 衣物用水 127升. 本模型可节约 18升.

第二章 规划与优化模型

实际问题中的优化模型可归结为:

min(or max) $z=f(x), x=(x_1,...,x_n)^T$

s.t. $g_i(x) \le 0$, (=0 或≥0) i = 1,...,m

x 称为决策变量, f(x)称为目标函数, $g_i(x)$ ≤0 称为约束条件. 具有这样形式的模型称为规划模型. 数学规划可分为:

- 1. 线性规划——f(x)、 $g_i(x)$ 均为线性函数,且 $x_i \ge 0$.
- 2. 非线性规划——f(x)或某 $g_i(x)$ 为非线性函数.
- 3. 整数规划—— $x_i \in \mathbb{Z}$.

第一节 奶制品的生产与销售

一奶制品加工厂用牛奶生产 A_1 , A_2 两种奶制品, 1 桶牛奶可以在设备甲上用 12 小时加工成 3 公斤 A_1 , 或者在设备乙上用 8 小时加工成 4 公斤 A_2 . 根据市场需求, 生产的 A_1 , A_2 全部能售出, 且每公斤 A_1 获利 24 元, 每公斤 A_2 获利 16 元. 现在加工厂每天能得到 50 桶牛奶的供应, 每天正式工人总的劳动时间为 480 小时, 并且设备甲每天至多能加工 100 公斤 A_1 , 设备乙的加工能力没有限制. 试为该厂制订一个生产计划, 使每天获利最大, 并进一步讨论以下 3 个问题:

- 1)若用 35 元可以买到 1 桶牛奶, 应否作这项投资?若投资, 每天最多购买多少桶牛奶?
- 2) 若可以聘用临时工人以增加劳动时间, 付给临时工人的工资最多是每小时几元?
- 3)由于市场需求变化、每公斤 A₁的获利增加到 30 元、应否改变生产计划?

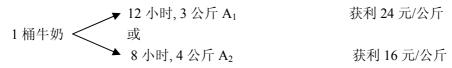
问题分析

空间层次

工厂级:根据外部需求和内部设备、人力、原料等条件,以最大利润为目标制订产品生产计划; 车间级:根据生产计划、工艺流程、资源约束及费用参数等,以最小成本为目标制订生产批量计划.

时间层次

若短时间内外部需求和内部资源等不随时间变化,可制订单阶段生产计划,否则应制订多阶段 生产计划.



每天: 50 桶牛奶, 时间 480 小时, 至多加工 100 公斤 A₁.

制订生产计划, 使每天获利最大

- •35 元可买到 1 桶牛奶, 要买吗. 若买, 每天最多买多少?
- 可聘用临时工人, 付出的工资最多是每小时几元?
- A₁的获利增加到 30 元/公斤, 应否改变生产计划?

建立线性规划模型(LP)

决策变量 x_1 桶牛奶生产 A_1 , 获利 $24 \times 3x_1$,

x₂桶牛奶生产 A₂, 获利 16×4x₂

目标函数 每天获利 $\max z=72x_1+64x_2$

约束条件 原料供应 $x_1+x_2 \le 50$

劳动时间 12x₁+8x₂≤480 加工能力 3x₁≤100

非负约束 $x_1,x_2 \ge 0$

模型分析与假设

- 比例性
- x_i 对目标函数的"贡献"与 x_i 取值成正比.
- x_i 对约束条件的"贡献"与 x_i 取值成正比.
- 可加性
- x_i 对目标函数的"贡献"与 x_i 取值无关.
- x_i 对约束条件的"贡献"与 x_i 取值无关.
- 连续性
- x_i 取值连续

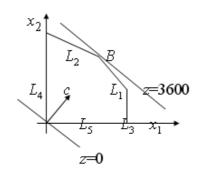
线性规划模型

- A₁, A₂每公斤的获利是与各自产量无关的常数.
- 每桶牛奶加工出 A₁, A₂的数量和时间是与各自产量无关的常数.
- A₁, A₂每公斤的获利是与相互产量无关的常数.
- 每桶牛奶加工出 A_1, A_2 的数量和时间是与相互产量无关的常数.
- 加工 A₁, A₂ 的牛奶桶数是实数.

模型求解

约束条件

1. 图解法



z=c (常数) 是等值线,目标函数和约束条件是线性函数,可行域为直线段围成的凸多边形,目标函数的等值线为直线,最优解一定在凸多边形的某个顶点取得.

结果: 在 B(20,30)点得到最优解.

图解法直观易行, 但当变量和约束条件较多时无法应用.

2. 用 LINDO 6.1 软件求解

max 72x1+64x2

st

2) x1+x2<50

- 3) 12x1+8x2<480
- 4) 3x1<100

end

求 LINDO 求解结果(要求作灵敏性分析):

LP OPTIMUM FOUND AT STEP

OBJECTIVE FUNCTION VALUE

	1)	3360.000	
VA	RIABLE	VALUE	REDUCED COST
	X1	20.000000	0.000000
	X2	30.000000	0.000000
	ROW	SLACK OR SURPLUS	DUAL PRICES
	2)	0.000000	48.000000
	3)	0.000000	2.000000
	4)	40.000000	0.000000
NIO	ITED ATIO	NIC 2	

NO. ITERATIONS= 2

RANGES IN WHICH THE BASIS IS UNCHANGED:

OBJ COEFFICIENT RANGES

VARIABLE	CURRENT	ALLOWA	BLE A	ALLOWABLE
	COEF	INCREASE	DECR	EASE
X1	72.000000	24.000000	8.000000	
X2	64.000000	8.000000	16.000000	
	RIG	HTHAND SIDE RA	ANGES	
ROW	CURRENT	ALLOWAE	BLE A	LLOWABLE
	RHS	INCREASE	DECR	EASE
2	50.000000	10.000000	6.666667	
3	480.000000	53.333332	80.000000	
4	100.000000	INFINITY	40.000000)

结果: 20 桶牛奶生产 A₁, 30 桶生产 A₂, 利润 3360 元.

结果解释

三种资源: 原料无剩余, 时间无剩余, 加工能力剩余 40.

"资源"剩余为零的约束为紧约束(有效约束),最优解下"资源"增加 1 单位时"效益"的增量——影子价格,原料增加 1 单位,利润增长 48 时间增加 1 单位,利润增长 2,加工能力增长不影响利润.

- 35 元可买到 1 桶牛奶, 要买吗? 35 <48, 应该买!
- 聘用临时工人付出的工资最多每小时几元? 2元! 最优解不变时目标函数系数允许变化范围(约束条件不变)
- *x*₁ 系数范围(64,96)
- x2系数范围(48,72)
- x_1 系数为 30×3=90, 在允许范围内
- A1 的获利增加到 30 元/公斤, 应否改变生产计划? 不变! 影子价格有意义时约束右端的允许变化范围 (目标函数不变) 原料最多增加 10

时间最多增加53

•35 元可买到 1 桶牛奶, 每天最多买多少? 最多买 10 桶!

思考题

前面给出的 A_1 , A_2 两种奶制品的生产条件、利润,及工厂的"资源"限制全都不变.为增加工厂的获利,开发了奶制品的深加工技术:用 2 小时和 3 元加工费,可将 1 公斤 A_1 加工成 0.8 公斤高级奶制品 B_1 ,也可将 1 公斤 A_2 加工成 0.75 公斤高级奶制品 B_2 ,每公斤月,能获利 44 元,每公斤 B_2 能获利 32 元.试为该厂制订一个生产销售计划,使每天的净利润最大,并讨论以下问题:

1)若投资 30 元可以增加供应 1 桶牛奶,投资 3 元可以增加 1 小时劳动时间,应否作这些投资?若每天投资 150 元,可赚回多少?

2)每公斤高级奶制品 B_1 , B_2 的获利经常有 10%的波动, 对制订的生产销售计划有无影响?若每公斤 B_1 的获利下降 10%, 计划应该变化吗?

第二节 自来水输送与货机装运

运输问题一般是指生产、生活物资从若干供应点运送到一些需求点,各种类型的货物装箱,由于受体积、重量等限制,如何搭配装载,使获利最高,或装箱数量最少,怎样安排输送方案使运费最小,或利润最大;

例1 自来水输送

某市有甲、乙、丙、丁四个居民区,自来水由 A,扫, C 三个水库供应.四个区每天必须得到保证的基本生活用水量分别为 30,70,10,10 千吨,但由于水源紧张,三个水库每天最多只能分别供应 50,60,50 千吨自来水.由于地理位置的差别,自来水公司从各水库向各区送水所需付出的引水管理费不同(见表,其中 C 水库与丁区之间没有输水管道),其他管理费用都是 450 元/千吨.根据公司规定,各区用户按照统一标准 900元/千吨收费.此外,四个区都向公司申请了额外用水量,分别为每天 50,70,20,40 千吨.该公司应如何分配供水量,才能获利最多?

为了增加供水量,自来水公司正在考虑进行水库改造,使三个水库每天的最大供水量都提高一倍,问那时供水方案应如何改变?公司利润可增加到多少?

引水管理费(元/千吨)	甲	乙	丙	1
A	160	130	220	170
В	140	130	190	150
С	190	200	230	/

问题分析

总供水量: 160< 总需求量: 120+180=300.

收入: 900 元/千吨、 总收入 900×160=144,000(元).

支出: 引水管理费.

模型建立 确定 3 个水库向 4 个小区的供水量

决策变量: 水库 i 向 j 区的日供水量为 $x_{ii}(x_{34}=0)$.

目标函数:

min $z=160x_{11}+130x_{12}+220x_{13}+170x_{14}+140x_{21}+130x_{22}+190x_{23}+150x_{24}+190x_{31}+200x_{32}+230x_{33}$ s.t.

 $x_{11}+x_{12}+x_{13}+x_{14}=50$

 $x_{21}+x_{22}+x_{23}+x_{24}=60$

 $x_{31} + x_{32} + x_{33} = 50$

 $30 \le x_{11} + x_{21} + x_{31} \le 80$

 $70 \le x_{12} + x_{22} + x_{32} \le 140$

 $10 \le x_{13} + x_{23} + x_{33} \le 30$

 $10 \le x_{14} + x_{24} \le 50$

模型求解

用 LINDO 求解以上模型(需要另外编写 LINGO 模型),得 OBJECTIVE FUNCTION VALUE

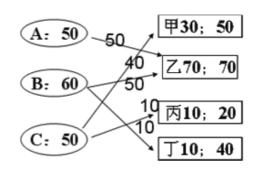
1) 24400.00

VARIABLE VALUE REDUCED

COST

X11	0.000000	30.000000
X12	50.000000	0.000000
X13	0.000000	50.000000
X14	0.000000	20.000000
X21	0.000000	10.000000
X22	50.000000	0.000000
X23	0.000000	20.000000
X24	10.000000	0.000000
X31	40.000000	0.000000
X32	0.000000	10.000000
X33	10.000000	0.000000

结果解释



引水管理费: 24400(元)

利润=总收入- 其它费用 - 引水管理费

=144000-72000-24400

=47600(元)

问题讨论

每个水库最大供水量都提高一倍.

总供水量(320) > 总需求量(300), 确定送水方案使利润最大.

利润= 收入(900) - 其它费用(450) -引水管理费

利润(元/千吨)	甲	Z	丙	丁
A	290	320	230	280
В	310	320	260	300
С	260	250	220	/

目标函数

 $\max z = 290x_{11} + 320x_{12} + 230x_{13} + 280x_{14} + 310x_{21} + 320x_{22} + 260x_{23} + 300x_{24} + 260x_{31} + 250x_{32} + 220x_{33}$

A: $x_{11}+x_{12}+x_{13}+x_{14}=50$ \longrightarrow $x_{11}+x_{12}+x_{13}+x_{14}\leq 100$

B, C 类似处理, 需求约束可以不变

求解结果

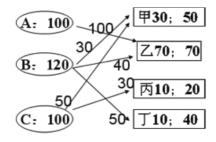
OBJECTIVE FUNCTION VALUE

1) 88700.00

VARIABLE VALUE REDUCED COST

X11	0.000000	20.000000
X12	100.000000	0.000000
X13	0.000000	40.000000
X14	0.000000	20.000000
X21	30.000000	0.000000
X22	40.000000	0.000000
X23	0.000000	10.000000
X24	50.000000	0.000000
X31	50.000000	0.000000
X32	0.000000	20.000000
X33	30.000000	0.000000

结果解释



总利润 88700(元)

例 2 货机装运

三个货舱最大载重(吨),最大容积(米3).飞机平衡条件

前仓:	中仓:	后仓:
10; 6800	16; 8700	8; 5300

三个货舱中实际载重必须与其最大载重成比例

	重量(吨)	空间(米 3 /吨)	利润(元/吨)
货物1	18	480	3100
货物 2	15	650	3800
货物3	23	580	3500
货物 4	12	390	2850

问如何装运, 使本次飞行获利最大?

模型假设

- 1、每种货物可以分割到任意小;
- 2、每种货物可以在一个或多个货舱中任意分布;
- 3、多种货物可以混装, 并保证不留空隙;
- 4、所给出的数据都是精确的,没有误差.

模型建立

决策变量

 x_{ij} --第 i 种货物装入第 j 个货舱的重量(吨) i=1,2,3,4, j=1,2,3 (分别代表前、中、后仓)

目标函数(利润)

 $\max Z = 3100(x11+x12+x13)+3800(x21+x22+x23)$

+3500(x31+x32+x33)+2850(x41+x42+x43)

约束条件

 x_{ij} --第i 种货物装入第j 个货舱的重量

 $x11+x21+x31+x41 \le 10$

 $x12+x22+x32+x42 \le 16$

 $x13+x23+x33+x43 \le 8$

 $480x11+650x21+580x31+390x41 \le 6800$

 $480x12+650x22+580x32+390x42 \le 8700$

 $480x13+650x23+580x33+390x43 \le 5300$

 $(x11 + x21 + x31 + x41) \; / \; 10 = (x12 + x22 + x32 + x42) \; / \; 16$

=(x13+x23+x33+x43)/8

 $x11+x12+x13 \le 18$

 $x21+x22+x23 \le 15$

 $x31+x32+x33 \le 23$

 $x41+x42+x43 \le 12$

模型求解

OBJECTIVE FUNCTION VALUE

1) 121515.8

VARIABLE VALUE REDUCED COST

X11	0.000000	400.000000
X12	0.000000	57.894737
X13	0.000000	400.000000
X21	10.000000	0.000000
X22	0.000000	239.473679
X23	5.000000	0.000000
X31	0.000000	0.000000
X32	12.947369	0.000000
X33	3.000000	0.000000
X41	0.000000	650.000000
X42	3.052632	0.000000
X43	0.000000	650.000000

货物 2: 前仓 10,后仓 5;

货物 3: 中仓 13, 后仓 3;

货物 4: 中仓 3.

最大利润约 121516 元

第三节 汽车生产与原油采购

例 1 汽车厂生产计划

一汽车厂生产小、中、大三种类型的汽车,已知各类型每辆车对钢材、劳动时间的需求,利润 以及每月工厂钢材、劳动时间的现有量如表中所示. 试制订月生产计划, 使工厂的利润最大.

进一步讨论: 由于各种条件限制,如果生产某一类型汽车,则至少要生产80辆,那么最优的生 产计划应作何改变.

	小型	中型	大型	现有量
钢材(吨)	1.5	3	5	600
劳动时间(小时)	280	250	400	60000
利润(万元)	2	3	4	

模型建立

设每月生产小、中、大型汽车的数量分别为 x_1, x_2, x_3 ,则

0.000000

$$\max z = 2x_1 + 3x_2 + 4x_3$$

s.t.

 $1.5x_1 + 3x_2 + 5x_3 \le 600$

 $280x_1 + 250x_2 + 400x_3 \le 60000$

$$x_1, x_2, x_3 \ge 0$$

模型求解

X1

OBJECTIVE FUNCTION VALUE

632.2581

VARIABLE VALUE REDUCED COST 64.516129

X2 167.741928 0.000000

0.000000 0.946237 X3

ROW SLACK OR SURPLUS DUAL PRICES

- 2) 0.000000 0.731183
- 3) 0.0000000.003226

结果为小数,怎么办?

- 1)舍去小数: 取 x_1 =64, x_2 =167, 算出目标函数值 z=629, 与 LP 最优值 632.2581 相差不大.
- 2)试探: 如取 x_1 =65, x_2 =167; x_1 =64, x_2 =168 等, 计算函数值 z, 通过比较可能得到更优的解. 但必 须检验它们是否满足约束条件. 为什么?
 - 3)模型中增加条件: x1, x2, x3 均为整数, 重新求解.

整数规划(Integer Programming, 简记 IP) 模型求解

 $\max z = 2x1 + 3x2 + 4x3$

s.t.

 $1.5x1+3x2+5x3 \le 600$

 $280x1+250x2+400x3 \le 60000$

x1,x2,x3 为非负整数

IP可用LINDO直接求解:

max 2x1+3x2+4x3

1.5x1+3x2+5x3<600

280x1+250x2+400x3<60000

end

gin 3

说明:

"gin 3"表示"前3个变量为整数", 等价于:

gin x1

gin x2

gin x3

求解结果

OBJECTIVE FUNCTION VALUE

1) 632.0000

VARIABLE VALUE REDUCED COST

X1 64.000000 -2.000000

X2 168.000000 -3.000000

X3 0.000000 -4.000000

最优解 x_1 =64, x_2 =168, x_3 =0, 最优值 z=632.

若生产某一类型汽车,则至少生产80辆,求生产计划.

方法 1: 修改模型为

 $\max z = 2x_1 + 3x_2 + 4x_3$

s.t.

 $1.5x_1+3x_2+5x_3 \le 600$

 $280x_1+250x_2+400x_3 \le 60000$

 $x_1, x_2, x_3=0$ 或 ≥ 80

将最后一个条件分解为 8 个 LP 子模型, 其中 3 个子模型应去掉, 然后逐一求解, 比较目标函数值, 再加上整数约束, 得最优解: x_1 =80, x_2 =150, x_3 =0, 最优值 z=610.

 $x_1=0, x_2=0, x_3\geq 80$

 $x_1=0, x_2\geq 80, x_3=0$

 $x_1=0, x_2\geq 80, x_3\geq 80$ (×)

 $x_1 \ge 80, x_2 = 0, x_3 = 0$

 $x_1 \ge 80, x_2 \ge 80, x_3 = 0$

 $x_1 \ge 80, x_2 = 0, x_3 \ge 80$

 $x_1 \ge 80, x_2 \ge 80, x_3 \ge 80$ (×)

 $x_1=0, x_2=0, x_3=0$ (×)

方法 2: 引入 0-1 变量, 化为整数规划.

 $x_1=0$ 或 ≥ 80 变为 $x_1 \leq M \cdot y_1, x_1 \geq 80y_1, y_1 \in \{0,1\}$

 $x_2=0$ 或 ≥ 80 变为 $x_2 \leq M \cdot y_2, x_2 \geq 80y_2, y_2 \in \{0,1\}$

 $x_3=0$ 或 ≥ 80 变为 $x_3 \leq M \cdot y_3, x_3 \geq 80y_3, y_3 \in \{0,1\}$

M 为大的正数, 可取 1000.

在 LINDO 中对 0-1 变量的限定:

int y1

int y2

int y3

解的结果

OBJECTIVE FUNCTION VALUE

1) 610.0000

VARIABLE VALUE REDUCED COST

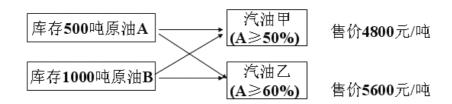
X1	80.000000	-2.000000
X2	150.000000	-3.000000
X3	0.000000	-4.000000
Y1	1.000000	0.000000
Y2	1.000000	0.000000
Y3	0.000000	0.000000

最优解同前

例 2 原油采购与加工

某公司用两种原油(A和B)混合加工成两种汽油(甲和乙). 甲、乙两种汽油含原油A的最低比例分别为50%和60%,每吨售价分别为4800元和5600元. 该公司现有原油A和B的库存量分别为500吨和1000吨,还可以从市场上买到不超过1500吨的原油A. 原油A的市场价为:购买量不超过500吨时的单价为10000元/吨;购买量超过500吨但不超过1000吨时,超过500吨的部分8000元/吨;购买量超过1000吨时,超过1000吨时,超过1000吨时,超过1000吨时,超过1000吨时,超过1000吨时,超过1000吨时,超过1000吨的部分6000元/吨. 该公司应如何安排原油的采购和加工?

问题分析



市场上可买到不超过 1500 吨的原油 A:

- 购买量不超过 500 吨时的单价为 10000 元/吨;
- 购买量超过 500 吨但不超过 1000 吨时, 超过 500 吨的部分 8000 元/吨;
- 购买量超过 1000 吨时, 超过 1000 吨的部分 6000 元/吨应如何安排原油的采购和加工?
- 利润: 销售汽油的收入- 购买原油 A 的支出.
- 难点: 原油 A 的购价与购买量的关系较复杂.

模型建立

决策变量: 设原油 A 的购买量为 x 吨, 用原油 A 生产汽油甲, 乙的数量分别为 x_{11}, x_{12} , 用原油 B 生产汽油甲, 乙的数量分别为 x_{21}, x_{22} ,

c(x) 是购买原油 A 的支出(千元)

目标函数: 利润(千元)

max z = 4.8(x11+x21)+5.6(x12+x22)-c(x)

c(x)如何表述?由已知条件得

$$c(x) = \begin{cases} 10x, & 0 \le x \le 500 \\ 8x + 1000, & 500 \le x \le 1000 \\ 6x + 3000, & 1000 \le x \le 1500 \end{cases}$$

约束条件: 库存 500 吨 A, 购买 x 吨 A, 库存 1000 吨 B.

$$x_{11} + x_{12} \le 500 + x$$
$$x_{21} + x_{22} \le 1000$$
$$x \le 1500$$

$$\frac{x_{11}}{x_{11} + x_{21}} \ge 0.5 \Rightarrow x_{11} \ge x_{21}$$
$$\frac{x_{12}}{x_{12} + x_{22}} \ge 0.6 \Rightarrow 2x_{12} \ge 3x_{22}$$

目标函数中 c(x)不是线性函数, 是非线性规划;

对于用分段函数定义的 c(x), 一般的非线性规划软件也难以输入和求解; 想办法将模型化简, 用现成的软件求解.

模型求解:

方法1

 x_1, x_2, x_3 是以价格 10, 8, 6(千元/吨)采购 A 的吨数

$$x = x_1 + x_2 + x_3$$
, $c(x) = 10x_1 + 8x_2 + 6x_3$

目标函数: $\max z = 4.8(x_{11} + x_{21}) + 5.6(x_{12} + x_{22}) - (10x_1 + 8x_2 + 6x_3)$

500 吨≤x≤1000 吨, 超过 500 吨的 8 千元/吨

增加约束

只有当以 10 千元/吨的价格购买 $x_1=500$ (吨)时, 才能以 8 千元/吨的价格购买 x_2 ,

$$(x_1 - 500)x_2 = 0$$
, $(x_2 - 500)x_3 = 0$, $0 \le x_1, x_2, x_3 \le 500$

非线性规划模型,可以用 LINGO 求解.

Model:

 $max = 4.8 \times x11 + 4.8 \times x21 + 5.6 \times x12 + 5.6 \times x22 - 10 \times x1 - 8 \times x2 - 6 \times x3$:

x11+x12 < x + 500;

x21+x22 < 1000;

0.5*x11 - 0.5*x21 > 0;

0.4*x12 - 0.6*x22 > 0;

x=x1+x2+x3;

(x1 - 500) * x2=0;

(x2 - 500) * x3=0;

x1 < 500;

x2 < 500;

x3 < 500;

x > 0;

x11 > 0;

x12 > 0;

x21 > 0;

x22 > 0;

x1 > 0;

 $x^2 > 0$;

x3 > 0;

end

求解结果

Objective value: 4800.000 Variable Value Reduced Cost

X11 500.0000

0.000000E+00

X21 500.0000 0.000000E+00 X12 0.0000000E+00 0.0000000E+00 X22 0.0000000E+00 0.000000E+00 X1 0.1021405E-13 10.00000 X2 0.0000000E+00 8.000000 X3 0.0000000E+00 6.000000

X = 0.0000000E+00 = 0.0000000E+00

用库存的 500 吨原油 A、500 吨原油 B 生产汽油甲,不购买新的原油 A,利润为 4,800 千元. LINGO 得到的是局部最优解,还能得到更好的解吗?

方法 2

 $y_1, y_2, y_3=1$ ——以价格 10, 8, 6(千元/吨)采购 A,

 x_1, x_2, x_3 ——以价格 10, 8, 6(千元/吨)采购 A 的吨数

增加约束

 $500y_2 \leqslant x_1 \leqslant 500y_1, 500y_3 \leqslant x_2 \leqslant 500y_2$

 $x_3 \le 500 y_3$

y₁,y₂,y₃是 0-1 变量,

 $y=0 \rightarrow x=0, x>0 \rightarrow y=1$

这是 0-1 线性规划模型,可用 LINDO 求解,购买 1000 吨原油 A,与库存的 500 吨原油 A 和 1000 吨原油 B 一起,生产汽油乙,利润为 5000 千元.

OBJECTIVE FUNCTION VALUE

1) 5000.000

VARIABLE VALUE REDUCED COST

Y1	1.000000	0.000000
Y2	1.000000	2200.000000
Y3	1.000000	1200.000000
X11	0.000000	0.800000
X21	0.000000	0.800000
X12	1500.000000	0.000000
X22	1000.000000	0.000000
X1	500.000000	0.000000
X2	500.000000	0.000000
X3	0.000000	0.400000
X	1000.000000	0.000000
优于方	7法1的结果	

第四节 接力队选拔和选课策略

本节介绍一类指派问题,指派问题一般指若干项任务分给一些候选人来完成,每人的专长不同, 完成每项任务取得的效益或需要的资源就不同,如何分派任务使获得的总效益最大,或付出的总资 源最少.

若干种策略供选择,不同的策略得到的收益或付出的成本不同,各个策略之间有相互制约关系,如何在满足一定条件下作出抉择,使得收益最大或成本最小.

例 1 混合泳接力队的选拔

5 名候选人的百米成绩如下表:

	甲	乙	丙	丁	戊
蝶泳	1'06"8	57"2	1'18"	1'10"	1'07"4
仰泳	1'15"6	1'06"	1'07"8	1'14"2	1'11"
蛙泳	1'27"	1'06"4	1'24"6	1'09"6	1'23"8
自由泳	58"6	53"	59"4	57"2	1'02"4

如何选拔队员组成 4×100 米混合泳接力队?

丁的蛙泳成绩退步到1'15"2; 戊的自由泳成绩进步到57"5, 组成接力队的方案是否应该调整?

问题分析

穷举法: 组成接力队的方案共有 5!=120 种.

0-1 规划模型, $c_{ii}(\emptyset)$ ——队员 i 第 i 种泳姿的百米成绩.

c_{ij}	<i>i</i> =1	<i>i</i> =2	<i>i</i> =3	<i>i</i> =4	<i>i</i> =5
<i>j</i> =1	66.8	57.2	78	70	67.4
<i>j</i> =2	75.6	66	67.8	74.2	71
<i>j</i> =3	87	66.4	84.6	69.6	83.8
<i>j</i> =4	58.6	53	59.4	57.2	62.4

若选择队员 i 参加泳姿 j 的比赛, 记 $x_{ii}=1$, 否则记 $x_{ii}=0$.

min
$$z = \sum_{i=1}^{4} \sum_{j=1}^{5} c_{ij} x_{ij}$$

每人最多入选泳姿之一,每种泳姿有且只有1人

$$\sum_{j=1}^{4} x_{ij} \le 1, \ i = 1, \dots, 5 \quad \sum_{i=1}^{5} x_{ij} = 1, \ j = 1, \dots, 4$$

模型求解

输入 LINDO 求解(中间省略了部分行,实际求解时是不可省略的).

MIN 66.8x11+75.6x12+87x13+58.6x14

+.....

+67.4x51+71 x52+83.8x53+62.4x54

SUBJECT TO

 $x11+x12+x13+x14 \le 1$

.

x41+x42+x43+x44 <=1

x11+x21+x31+x41+x51=1

.

x14+x24+x34+x44+x54=1

END

INT 20

最优解: $x_{14} = x_{21} = x_{32} = x_{43} = 1$, 其它变量为 0, 成绩为 253.2(秒)=4'13"2. 方案为: 甲~ 自由泳、乙~ 蝶泳、丙~ 仰泳、丁~ 蛙泳.

讨论:丁蛙泳 c_{43} =69.6→75.2、戊自由泳 c_{54} =62.4→57.5、方案是否调整? (敏感性分析)

IP 规划一般没有与 LP 规划相类似的理论, LINDO 输出的敏感性分析结果通常是没有意义的.

将 c_{43} , c_{54} 的新数据重新输入模型,用 LINDO 求解. 最优解: $x_{21}=x_{32}=x_{43}=x_{51}=1$,成绩为 4'17"7. 方案为: 乙~ 蝶泳、丙~ 仰泳、丁~ 蛙泳、戊~ 自由泳.

例 2 选课策略

某专业课程设置如下表所示:

课号	课名	学分	所属类别	先修课要求
1	微积分	5	数学	
2	线性代数	4	数学	
3	最优化方法	4	数学; 运筹学	微积分; 线性代数
4	数据结构	3	数学; 计算机	计算机编程
5	应用统计	4	数学; 运筹学	微积分; 线性代数
6	计算机模拟	3	计算机; 运筹学	计算机编程
7	计算机编程	2	计算机	
8	预测理论	2	运筹学	应用统计
9	数学实验	3	运筹学; 计算机	微积分; 线性代数

要求至少选两门数学课、三门运筹学课和两门计算机课,为了选修课程门数最少,应学习哪些课程?选修课程最少,且学分尽量多,应学习哪些课程?

0-1 规划模型

决策变量

x=1 是选修课号 i 的课程(x=0 表示不选).

目标函数

选修课程总数最少

$$\min \ z = \sum_{i=1}^{9} x_i$$

约束条件

最少2门数学课

 $x_1 + x_2 + x_3 + x_4 + x_5 \ge 2$

3 门运筹学课

 $x_3 + x_5 + x_6 + x_8 + x_9 \ge 3$

2 门计算机课

 $x_4 + x_6 + x_7 + x_9 \ge 2$

先修课要求 $x_3=1$ 必有 $x_1=x_2=1$ $\rightarrow x_3 \leqslant x_1, x_3 \leqslant x_2 \rightarrow 2x_3-x_1-x_2 \leqslant 0$.

$$x_4 \leqslant x_7 \rightarrow x_4 - x_7 \leqslant 0.$$

$$2x_5-x_1-x_2 \leq 0$$

 $x_6 - x_7 \le 0$

 $x_8 - x_5 \le 0$

 $2x_9-x_1-x_2 \leq 0$

模型求解(LINDO)

最优解: $x_1 = x_2 = x_3 = x_6 = x_7 = x_9 = 1$, 其它为 0, 6 门课程, 总学分 21.

讨论: 选修课程最少, 学分尽量多, 应学习哪些课程?

课程最少

$$\min \ z = \sum_{i=1}^{9} x_i$$

学分最多

max
$$w = 5x_1 + 4x_2 + 4x_3 + 3x_4 + 4x_5 + 3x_6 + 2x_7 + 2x_8 + 3x_9$$

两目标(多目标)规划—— $\min \{z,-w\}$.

多目标优化的处理方法: 化成单目标优化.

以课程最少为目标,不管学分多少,则最优解如上,6门课程,总学分21。 以学分最多为目标,不管课程多少,则最优解显然是选修所有9门课程.

多目标规划

在课程最少的前提下以学分最多为目标. 增加约束

$$\sum_{i=1}^{9} x_i = 6$$

以学分最多为目标求解,最优解: $x_1 = x_2 = x_3 = x_5 = x_7 = x_9 = 1$,其它为 0,总学分由 21 增至 22.

注意: 最优解不唯一!

可将 $x_9 = 1$ 易为 $x_6 = 1$.

LINDO 无法告诉优化问题的解是否唯一.

课号	课名	学分
1	微积分	5
2	线性代数	4
3	最优化方法	4
4	数据结构	3
5	应用统计	4
6	计算机模拟	3
7	计算机编程	2
8	预测理论	2
9	数学实验	3

对学分数和课程数加权形成一个目标, 如三七开.

min
$$Y = \lambda_1 z - \lambda_2 w = 0.3z - 0.7w$$

其中
$$z = \sum_{i=1}^{9} x_i$$
, $w = 5x_1 + 4x_2 + 4x_3 + 3x_4 + 4x_5 + 3x_6 + 2x_7 + 2x_8 + 3x_9$

最优解: x₁ = x₂ = x₃ = x₄= x₅ = x₆ = x₇ = x₉ = 1, 其它为 0; 总学分 28

课号	课名	学分
1*	微积分	5
2*	线性代数	4
3*	最优化方法	4
4*	数据结构	3
5*	应用统计	4
6*	计算机模拟	3
7*	计算机编程	2
8	预测理论	2
9*	数学实验	3

讨论与思考

$$\min \ Y = \lambda_1 z - \lambda_2 w, \ \lambda_1 + \lambda_2 = 1, \ 0 \le \lambda_1, \lambda_2 \le 1$$

其中
$$z = \sum_{i=1}^{9} x_i$$
 , $w = 5x_1 + 4x_2 + 4x_3 + 3x_4 + 4x_5 + 3x_6 + 2x_7 + 2x_8 + 3x_9$ $\lambda_1 < \frac{2}{3}$ 最优解与 $\lambda_1 = 0$, $\lambda_2 = 1$ 的结果相同——学分最多. $\lambda_1 < \frac{3}{4}$ 最优解与 $\lambda_1 = 1$, $\lambda_2 = 0$ 的结果相同——课程最少.

第五节 饮料厂的生产与检修

例 1 饮料厂的生产与检修计划

某饮料厂生产一种饮料用以满足市场需求. 该厂销售科根据市场预测,已经确定了未来四周该饮料的需求量. 计划科根据本厂实际情况给出了未来四周的生产能力和生产成本,如表中所示. 每周当饮料满足需求后有剩余时,要支出存贮费,为每周每千箱饮料 0.2 千元. 问应如何安排生产计划,在满足每周市场需求的条件下,使四周的总费用(生产成本与存贮费之和)最小?

如果工厂必须在未来四周的某一周中安排一次设备检修, 检修将占用当周 15 千箱的生产能力, 但会使检修以后每周的生产能力提高 5 千箱, 则检修应该安排在哪一周.

周次	需求量(千箱)	生产能力(千箱)	成本(千元/千箱)
1	15	30	5.0
2	25	40	5.1
3	35	45	5.4
4	25	20	5.5
合计	100	135	

问题分析

- 除第 4 周外每周的生产能力超过每周的需求;
- 生产成本逐周上升;
- 前几周应多生产一些.

模型假设

- 饮料厂在第1周开始时没有库存;
- 从费用最小考虑, 第 4 周末不能有库存;
- 周末有库存时需支出一周的存贮费;
- 每周末的库存量等于下周初的库存量.

模型建立

决策变量

 $x_1 \sim x_4$: 第 1~4 周的生产量.

y₁~y₃: 第 1~3 周末库存量.

存贮费: 0.2 (千元/周•千箱).

目标函数

min $z=5x_1+5.1x_2+5.4x_3+5.5x_4+0.2(y_1+y_2+y_3)$

约束条件

 $x_1-y_1=15$

 $x_2 + y_1 - y_2 = 25$

 $x_3 + y_2 - y_3 = 35$

 $x_4 + y_3 = 25$

 $x_1 \leq 30, x_2 \leq 40$

 $x_3 \leq 45, x_4 \leq 20$

 $x_1, x_2, x_3, x_4, y_1, y_2, y_3 \ge 0$

模型求解

LINDO 求解,最优解: $x_1 \sim x_4$: 15, 40, 25, 20; $y_1 \sim y_3$: 0, 15, 5.

周次	需求	产量	库存	能力	成本
1	15	15	0	30	5.0
2	25	40	15	40	5.1
3	35	25	5	45	5.4
4	25	20	0	20	5.5

4周生产计划的总费用为528(千元).

检修计划

在 4 周内安排一次设备检修, 占用当周 15 千箱生产能力, 能使检修后每周增产 5 千箱, 检修应排在哪一周?

检修安排在任一周均可.

0-1 变量 w_t: w_t=1~ 检修安排在第 t 周(t=1,2,3,4)

约束条件

能力限制

 $x_1 \le 30$ $x_1 + 15w_1 \le 30$ $x_2 \le 40$ $x_2 + 15w_2 \le 40 + 5w_1$

 $x_3 \le 45$ $x_3 + 15w_3 \le 45 + 5w_2 + 5w_1$

 $x_4 \le 20$ \longrightarrow $x_4+15w_4 \le 20+5w_3+5w_2+5w_1$

目标函数不变,增加约束条件: 检修1次

 $w_1+w_2+w_3+w_4=1$

LINDO 求解

最优解: $w_1=1, w_2, w_3, w_4=0$;

 $x_1 \sim x_4$: 15, 45, 15, 25;

 $y_1 \sim y_3$: 0, 20, 0.

总费用由 528 降为 527(千元)

检修所导致的生产能力提高的作用,需要更长的时间才能得到充分体现.

例 2 饮料的生产批量问题

饮料厂使用同一条生产线轮流生产多种饮料. 若某周开工生产某种饮料, 需支出生产准备费 8 千元.

某种饮料 4 周的需求量、生产能力和成本

周次	需求量(千箱)	生产能力(千箱)	成本(千元/千箱)
1	15	30	5.0
2	25	40	5.1
3	35	45	5.4
4	25	20	5.5
合计	100	135	

存贮费: 每周每千箱饮料 0.2 (千元).

安排生产计划,满足每周的需求,使4周总费用最小.

符号假设

 $ct \sim$ 时段 t 生产费用(元/件);

 $ht \sim$ 时段 $t(\bar{x})$ 库存费(元/件)

st~时段 t 生产准备费(元);

 $dt \sim$ 时段 t 市场需求(件);

Mt~时段 t 生产能力(件).

决策变量

 $xt \sim$ 时段 t 生产量;

yt~时段 $t(\pi)$ 库存量

 $wt = 1 \sim$ 时段 t 开工生产

(wt =0~不开工).

假设初始库存为0,制订生产计划,满足需求,并使T个时段的总费用最小.

目标函数

min
$$z = \sum_{t=1}^{T} (s_t w_t + c_t x_t + h_t y_t)$$

约束条件

$$y_{t-1} + x_t - y_t = d_t$$

$$w_t = \begin{cases} 1, & x_t > 0 \\ 0, & x_t = 0 \end{cases}, x_t \le M_t$$

$$y_0 = y_T = 0, x_t, y_t \ge 0, t = 1, ..., T$$

这是一个混合 0-1 规划模型, 用 LINDO 求解

最优解: x1~x4:15,40,45,0; 总费用:554.0(千元)

思考题

1. 钢管下料问题

原料钢管:

每根 19 米

客户需求:

4米50根

6米20根

8米15根

问题 1. 如何下料最节省? 节省的标准是什么?

问题 2. 客户增加需求:

5米10根

由于采用不同切割模式太多,会增加生产和管理成本,规定切割模式不能超过3种.如何下料最节省?

2. 发动机生产计划

某厂向用户提供发动机. 合同规定,第一、二、三季度末分别交货 40 台、60 台、80 台. 每季度的生产费用为 $f(x)=ax+bx^2$ (元),其中 x 是该季度生产的台数.若交货后有剩余,可用于下季度交货,但需支付存储费,每台每季度 c 元.已知工厂每季度最大生产能力为 100 台,第一季度开始时无存货,设 a=50,b=0.2,c=4,问工厂应如何安排生产计划,才能既满足合同又使总费用最低?讨论 a,b,c 变化对计划的影响,并作出合理的解释.

第三章 微分方程模型

建立微分方程模型要对研究对象作具体分析. 一般有以下三种方法: ①根据规律建模; ②用微元法建模; ③用模拟近似法建模.

第一节 根据规律建模

在数学、力学、物理、化学等学科中已有许多经过实践检验的规律和定律,如牛顿运动定律、基尔霍夫电流及电压定律、物质的放射规律、曲线的切线性质等,这些都涉及到某些函数的变化率.我们就可以根据相应的规律,列出常微分方程.下面以目标跟踪问题为例介绍.

设位于坐标原点的甲舰向位于x轴上点 A(1,0)处的乙舰发射导弹,导弹始终对准乙舰. 如果乙舰以最大的速度 V_0 沿平行于y轴的直线行驶,导弹的速度是 5 V_0 ,求导弹运行的曲线.又乙舰行驶多远时,导弹将它击中?

解: 设导弹轨迹为 y=y(x), 经过时间 t, 导弹位于点 P(x,y), 乙舰位于点 $Q(1,V_0t)$. 由于导弹头始终对准乙舰, 故此时 PQ 就是曲线 y(x)在点 P 处的切线, 因此, 由 $y'=\frac{V_0t-y}{1-x}$ 得

$$V_0 t = (1 - x)y' + y$$
.

又因为弧 OP 的长度为 5|AO|, 即

$$\int_0^x \sqrt{1 + {y'}^2} \, dx = 5V_0 t$$

所以

$$(1-x)y' + y = \frac{1}{5} \int_0^x \sqrt{1+{y'}^2} dx$$

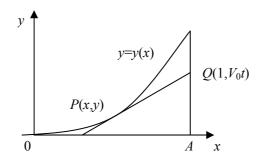
整理得

$$(1+x)y'' = \frac{1}{5}\sqrt{1+{y'}^2} ,$$

并有 y(0) = 0, y'(0) = 0,解得

$$y = -\frac{5}{8}(1-x)^{\frac{4}{5}} + \frac{5}{12}(1-x)^{\frac{6}{5}} + \frac{5}{24}$$

当 x=1 时, $y=\frac{5}{24}$,即当乙舰行到 $(1,\frac{5}{24})$ 处被击中, $t=\frac{y}{V_0}=\frac{5}{24V_0}$.



第二节 微元法建模

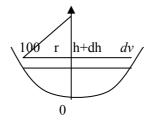
微元法建模实际上是寻求一些微元之间的关系式. 与第一种方法不同之处在于这里不是直接对未知函数及其导数应用规律和定理来求关系式, 而是对某些微元来应用规律. 以容器漏水问题为例.

有高为1米的半球形容器,水从它的底部小孔流出. 小孔横截面为 1cm^2 . 开始时容器内盛满了水,求水从小孔流出过程中容器里水面的高度 h(水面与孔口中心距离)随时间 t 变化的规律.

解:由流体力学知识知道,水从孔口流出的流量 Q 可用下列公式计算:

$$Q = \frac{dv}{dt} = 0.62 S \sqrt{2gh} ,$$

其中 0.62 为流量系数, S 为孔口横截面积. 现 S=1cm², 故 $\frac{dv}{dt}=0.62\sqrt{2gh}$.



另一方面,设在 $[t,t+\Delta t]$ 内,水面高度由h降至h+dh(dh<0),则

$$dv = -\pi r^2 dh$$

其中 r 是时刻 t 的水面半径. 因为 $r = \sqrt{100^2 - (100 - h)^2} = \sqrt{200h - h^2}$, 所以 $dv = -\pi (200h - h^2)dh$

于是, $0.62\sqrt{2gh}dt = -\pi(200h - h^2)dh$. 由此得

$$\frac{dt}{dh} = \frac{\pi}{0.62\sqrt{2g}} \left(h^{\frac{3}{2}} - 200h^{\frac{1}{2}}\right)$$

满足: $h|_{t=0}=100$. 解得

$$t = \frac{\pi}{4.65\sqrt{2g}} (7 \times 10^5 - 10^3 h^{\frac{3}{2}} + 3h^{\frac{5}{2}})$$

此即容器内水面高度 h 与时间 t 之间的函数关系式.

第三节 模拟近似法建模

在社会科学、生物学、医学、经济学等学科的实践中,常常要用模拟近似法来建立微分方程模型. 这是因为,这些学科中的一些现象的规律我们还不是很清楚,即使有所了解也并不全面,因此,要用数学模型进行研究只能在不同的假设下去模拟实际的现象. 然后再把解的结果同实际情况作对比. 以交通管理问题为例.

在交通十字路口,都会设置红绿灯.为了让那些正行驶在交叉路口或离交叉路口太近而无法停

下的车辆通过路口,红绿灯转换中间还要亮起一段时间的黄灯.对于一位驶近交叉路口的驾驶员来说,万万不可处于这样的进退两难的境地:要安全停车则离路口太近;要想在红灯亮之前通过路口又觉得太远.那么,黄灯应亮多长时间才最为合理呢?

解: 各段时间应该满足以下关系:

黄灯状态应持续的时间=驾驶员反应时间+车通过交叉路口时间+通过刹车距离的时间.

设 v_0 一表示法定速度,I一交叉路口宽度,L一典型车身长度。则通过路口的时间为 $\frac{I+L}{v_0}$,(车尾

通过路口). 下面计算刹车距离.

设 w—为汽车的重量, μ —磨擦系数,则摩擦力= μw ,汽车在停车过程中,行驶距离 x 与时间 t 的关系可由下面微分方程求得

$$\frac{w}{g}\frac{d^2x}{dt^2} = -\mu w \quad (F=ma).$$

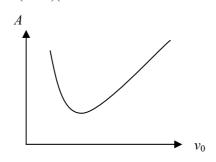
满足: $x\big|_{t=0}=0, x'\big|_{t=0}=v_0$, 于是刹车距离就是直到速度 v=0 时汽车驶过的距离, 由上式得

$$x = -\frac{1}{2}\mu g t^2 + v_0 t$$
. 令 $x' = 0$,所以刹车所用时间 $t_0 = \frac{v_0}{\mu g}$,刹车距离 $x(t_0) = \frac{{v_0}^2}{2\mu g}$.

由上面得黄灯状态时间为

$$A = \frac{x(t_0) + I + L}{v_0} + T = \frac{v_0}{2\mu g} + \frac{I + L}{v_0} + T$$

其中T是驾驶员反应时间,A, v_0 关系(如图)(即黄灯周期与法定速度的关系).



假设 T=1s, L=4.5m, I=9m, 另外, 我们取具有代表性的 μ =0.2, 当 ν_0 =45, 60, 80km/h 时, 黄灯时间如下表所示.

$v_0(\text{km/h})$	A(s)	经验法的值(s)
45	5.27	3
60	6.06	4
80	7.28	5

经验法的结果一律比我们预测的黄灯状态短些. 这使人想起, 许多交叉路口红绿灯的设计可能 使车辆在绿灯转为红灯时正处于交叉路口.

第四节 微分方程模型实例

例 1. 最优捕鱼策略

为了保护人类赖以生存的自然环境,可再生资源(如渔业、林业资源)的开发必须适度.一种合理、简化的策略是,在实现可持续收获的前提下,追求最大产量或最佳效益.

考虑对某种鱼(鲥鱼)的最优捕捞策略:

假设这种鱼分 4 个年龄组: 称 1 龄鱼, ..., 4 龄鱼. 各年龄组每条鱼的平均重量分别为 5.07, 11.55, 17.86, 22 .99(克); 各年龄组鱼的自然死亡率均为 0.8(1/年); 这种鱼为季节性集中产卵繁殖, 平均每条 4 龄鱼的产卵量为 1.109×10^5 (个), 3 龄鱼的产卵量为这个数的一半, 2 龄鱼和 1 龄鱼不产卵, 产卵和孵化期为每年的最后 4 个月; 卵孵化并成活为 1 龄鱼, 成活率(1 龄鱼条数与产卵总量 n 之比)为 $1.22 \times 10^{11}/(1.22 \times 10^{11} + n)$.

渔业管理部门规定,每年只允许在产卵孵化期前的 8 个月内进行捕捞作业.如果每年投入的捕捞能力(如渔船数、下网次数等)固定不变,这时单位时间捕捞量将与各年龄组鱼群条数成正比,比例系数不妨称捕捞强度系数.通常使用 13mm 网眼的拉网,这种网只能捕捞 3 龄鱼和 4 龄鱼,其两个捕捞强度系数之比为 0.42:1. 渔业上称这种方式为固定努力量捕捞.

- (1)建立数学模型分析如何实现可持续捕获(即每年开始捕捞时渔场中各年龄组鱼群条数不变),并且在此前提下得到最高的年收获量(捕捞总重量).
- (2)某渔业公司承包这种鱼的捕捞业务 5 年, 合同要求 5 年后鱼群的生产能力不能受到太大破坏. 已知承包时各年龄组. 鱼群的数量分别为: 122, 29.7, 10.1, 3.29(×10⁹ 条), 如果仍用固定努力量的捕捞方式. 该公司应采取怎样的策略才能使总收获量最高.

问题分析

要求研究的问题是: 对某种鱼的最优捕捞策略.

1、鱼的情况

具体数据如下表:

i	$m_i(g)$	r(1/年)	<i>u_i</i> (个/条)
1	5.07	0.8	0
2	11.55	0.8	0
3	17.86	0.8	
4	22.99	0.8	

其中, i 表示 i 龄鱼, m_i 表示 i 龄鱼的质量, r 表示 i 龄鱼的自然死亡率, u_i 表示平均每条 i 龄鱼的产卵量. 如果每年投入的捕捞能力(如渔船数、下网次数)不变, 这时单位时间捕捞量将与 i 成正比, 比例系数(称捕捞强度系数)为 k_i 使用 13mm 网眼的拉网, 这种网只能捕捞 3、4 龄鱼, 其两个捕捞强度系数之比为 k_3 : k_4 =0.42:1, k_1 = k_2 =0. 渔业上称这种方式为固定努力量捕捞.

2、基本假设

假设 I: 一年中, 鱼的产卵是集中在 8 月底一次性完成, 捕捞工作只在前 8 个月进行.

假设 II:各龄鱼(不包括 4 龄鱼)只在年末瞬时才长大一岁. 鱼卵在年终才孵化完毕, 成为 1 龄鱼. 这样在计算产卵量时, 3, 4 龄鱼的条数为 t=8/12.

3、应解决的问题

1)建立数学模型分析如何实现可持续捕捞(即每年开始捕捞时渔场中各年龄组鱼的条数不变), 且在此前提下得到最高的年收获量(捕捞总量).

2)某渔业公司承包这种鱼的捕捞业务 5 年, 合同要求 5 年后鱼群的生产能力不能受到太大破坏. 已知承包时各年龄组鱼群的数量分别为 x_1, x_2, x_3, x_4 , 如果仍用固定努力量的捕捞方式, 该公司应采取怎样的策略才能使总收获量最高.

记号和约定

 $x_i(t)$: i 龄鱼在 t 时刻的数量(t 以年为单位, i=1,2,3,4);

 p_i : i 龄鱼的捕捞量(i=3,4);

M: 捕捞总质量;

Q: 每年的产卵中能孵化成1龄鱼的数量;

N: 每年的产卵量

模型的建立

由于鱼的数量随时间变化,可视 $x_i(t)$ 为连续函数,它的变化与时间 t、自然死亡率 r、单位时间捕捞量 k_i 、卵的成活率有关.

模型I

定义单位死亡率

$$\frac{dx_i}{dt} = -rx_i, r = 0.8$$

单位时间捕捞量

$$\frac{dp_i}{dt} = k_i x_i, \quad k_3 : k_4 = 0.42 : 1, \quad k_1 = k_2 = 0$$

则捕捞时满足

$$\frac{dx_i}{dt} = -(r + k_i)x_i$$

对各龄鱼存在以下方程(令 $k=k_4$, 则 $k_3=0.42k$)

$$\frac{dx_1(t)}{dt} = -0.8x_1, t \in [0,1],
\frac{dx_2(t)}{dt} = -0.8x_2, t \in [0,1],
\begin{cases}
\frac{dx_3(t)}{dt} = -(0.8 + 0.42k)x_3, t \in [0, \frac{8}{12}],
\frac{dx_3(t)}{dt} = -0.8x_3, t \in [\frac{8}{12}, 1],
\end{cases}$$

$$\begin{cases}
\frac{dx_4(t)}{dt} = -(0.8 + k)x_4, t \in [0, \frac{8}{12}],
\frac{dx_4(t)}{dt} = -0.8x_4, t \in [\frac{8}{12}, 1],
\end{cases}$$

由此可解得

$$x_{1}(1) = \exp(-0.8)x_{1}(0),$$

$$x_{2}(1) = \exp(-0.8)x_{2}(0),$$

$$x_{3}(\frac{8}{12}) = \exp(-(0.8 + 0.42k) \times \frac{2}{3})x_{3}(0),$$

$$x_{4}(\frac{8}{12}) = \exp(-(0.8 + k) \times \frac{2}{3})x_{4}(0),$$

$$x_{3}(1) = \exp(-(0.8 + 0.28k))x_{3}(0),$$

$$x_{4}(1) = \exp(-(0.8 + \frac{2k}{3})x_{4}(0),$$

收获量为

$$p_3 = \int_0^{\frac{8}{12}} 0.42kx_3(t)dt, \ p_4 = \int_0^{\frac{8}{12}} kx_4(t)dt,$$

模型II

要实现持续收获,即每年初,各年龄组鱼群数量不变,同时须满足以下等式(M 为捕捞总质量,N 为卵量,Q 为存活量),由此可建立模型:

$$\max M = p_3 m_3 + p_4 m_4$$

s.t.
$$\begin{cases} x_1(0) = Q, \\ x_2(0) = x_1(1), \\ x_3(0) = x_2(1), \\ x_4(0) = x_3(1) + x_4(1). \end{cases}$$

其中 $N = (1.109 \times \frac{1}{2} x_3 + 1.109 x_4) \times 10^5$, $Q = 1.22 \times 10^{11} N / (1.22 \times 10^{11} + N)$.

利用约束关系,将 $x_1(0)$,..., $x_4(0)$ 均用k来表示,从而得出M关于k的一元函数关系:

$$M = m_3 p_3 + m_4 p_4,$$

$$\begin{split} p_3 &= 0.42kx_3(0)(1 - \exp(-(0.8 + 0.42k) \times \frac{2}{3})/(0.8 + 0.42k) \,, \\ p_4 &= kx_4(0)(1 - \exp(-(0.8 + k) \times \frac{2}{3})/(0.8 + k) \,, \\ x_3(0) &= 0.2463 \times 10^{11} - 3.7504 \times 10^6 \times \exp(0.28k) \times (2.226 - \exp(-2 \times \frac{k}{3})) \,, \\ /(2.226 + \exp(-2 \times \frac{k}{3})) \,, \end{split}$$

$$x_4(0) = \exp(-0.28k) \times 10^{11} / (9.0272 - 4.0598 \times \exp(-2 \times \frac{k}{3}))$$
$$-1.22 \times 10^6 / (0.7241 + 0.3252 \times \exp(-2 \times \frac{k}{3}))$$

用计算机搜索, 得出近似解, 结果为:

年最大收获量:
$$M^*(0) = 3.88685 \times 10^{11}(g)$$

最佳捕捞强度系数: $K^* = 17.7600(g)$

此时

$$x_1^*(0) = 1.193 \times 10^{11}, \quad x_2^*(0) = 5.360 \times 10^{10},$$

 $x_3^*(0) = 2.409 \times 10^{10}, \quad x_4^*(0) = 7.501 \times 10^7,$
 $Q^*(0) = 1.193 \times 10^{11}.$

四、模型结果及实用性讨论

当 $k \in [0, 31.38]$ 时,鱼场可实现持续捕捞,即满足了持续捕捞条件 $M \ge 0$. 在此前提下取得了最优的捕捞系数: 3 龄鱼的捕捞系数为 7.4592, 4 龄鱼的捕捞系数为 17.76, 年最大捕捞量为:3.88685× 10^{11} (克).

例 2. 存贮模型

某厂生产若干种产品. 轮换生产时因更换设备要付准备费; 产量大于需求时因积压资金要付贮存费.

今已知某产品的日需求量为 100 件, 生产准备费 5000 元, 贮存费每日每件 1 元. 试安排该产品的生产计划, 即多少天生产一次(生产周期), 每次产量多少, 使总费用最小. 设该厂生产能力非常大,即所需数量可在很短时间内产出.

要求: 不只是回答问题, 而且要建立生产周期、产量与需求量、准备费、贮存费之间的关系.

问题分析与思考

- 日需求 100 件, 生产准备费 5000 元, 贮存费每日每件 1 元.
- 每天生产一次、每次 100 件、无贮存费、准备费 5000 元、故每天费用为 5000 元.
- 10 天生产一次,每次 1000 件, 贮存费 900+800+···+100 =4500 元,准备费 5000 元,总计 9500 元,平均每天费用为 950 元.
- 50 天生产一次,每次 5000 件, 贮存费 4900+4800+···+100 =122500 元,准备费 5000 元,总计 127500 元.平均每天费用为 2550 元.
- 周期短,产量小→ 贮存费少,准备费多.
- 周期长,产量大→ 准备费少,贮存费多.

是否存在最佳的周期和产量, 使总费用(二者之和)最小?

这是一个优化问题, 关键在建立目标函数. 显然不能用一个周期的总费用作为目标函数.

目标函数——每天总费用的平均值.

模型假设

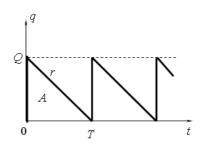
- 1. 产品每天的需求量为常数 r;
- 2. 每次生产准备费为 c_1 , 每天每件产品贮存费为 c_2 ;
- 3. T天生产一次(周期为 T), 每次生产 Q 件,且当贮存 量降到零时,Q 件产品立即生产出来(生产时间不计):
- 4. 为方便起见, 时间和产量都作为连续量处理.

建模目的

设 r, c_1, c_2 已知, 求 T, Q, 使每天总费用的平均值最小.

模型建立

离散问题连续化



将贮存量表示为时间的函数 q(t), t=0 生产 Q 件,贮存量 q(0)=Q, q(t) 以需求 r 的速率递减,直到 q(T)=0. 由此得

Q=rT

一周期贮存费

$$c_2 \int_0^T q(t)dt = c_2 A$$

一周期总费用

$$\overline{C} = c_1 + c_2 \frac{Q}{2} T = c_1 + c_2 \frac{rT^2}{2}$$

每天总费用平均值(目标函数)

$$C(T) = \frac{\overline{C}}{T} = \frac{c_1}{T} + \frac{c_2 rT}{2}$$

模型求解

求
$$T$$
 使 $C(T) = \frac{c_1}{T} + \frac{c_2 rT}{2} \rightarrow \min$.
由 $\frac{dC}{dT} = 0$ 有

$$T = \sqrt{\frac{2c_1}{rc_2}}, \quad Q = rT = \sqrt{\frac{2c_1r}{c_2}}.$$

模型分析

$$c_1 \uparrow \Rightarrow T, Q \uparrow$$
, $c_2 \uparrow \Rightarrow T, Q \downarrow$, $r \uparrow \Rightarrow T \downarrow, Q \uparrow$.

模型应用

经济批量订货公式(EOQ 公式)

用于订货、供应、存贮情形

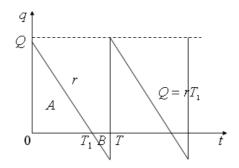
每天需求量r, 每次订货费为 c_1 ,每天每件贮存费为 c_2 , T天订货一次(周期T), 每次订货Q件, 且当贮存量降到 零时, Q件立即到货.

$$T = \sqrt{\frac{2c_1}{rc_2}}$$
 , $Q = rT = \sqrt{\frac{2c_1r}{c_2}}$

这是不允许缺货的存贮模型

问: 为什么不考虑生产费用? 在什么条件下才不考虑?

允许缺货的存贮模型



当贮存量降到零时仍有需求 r, 出现缺货, 造成损失.

原模型假设: 贮存量降到零时 Q 件立即生产出来(或立即到货).

现假设: 允许缺货, 每天每件缺货损失费 c3, 缺货需补足.

周期 T, $t=T_1$ 贮存量降到零.

T一周期贮存费

$$c_2 \int_0^{T_1} q(t)dt = c_2 A$$

一周期缺货费

$$c_3 \int_{T_1}^{T} |q(t)| dt = c_3 B$$

一周期总费用

$$\overline{C} = c_1 + c_2 \frac{QT_1}{2} + c_3 \frac{r(T - T_1)^2}{2}$$

每天总费用平均值(目标函数)

$$C(T,Q) = \frac{\overline{C}}{T} = \frac{c_1}{T} + \frac{c_2 Q^2}{2rT} + \frac{c_3 (rT - Q)^2}{2rT}$$

求 T,Q 使

$$C(T,Q) \rightarrow \min$$

 $\frac{\partial C}{\partial T} = 0$, $\frac{\partial C}{\partial Q} = 0$, 为与不允许缺货的存贮模型相比, T 记作 T', Q 记作 Q'.

$$T' = \sqrt{\frac{2c_1}{rc_2} \frac{c_2 + c_3}{c_3}}, \quad Q' = \sqrt{\frac{2c_1r}{c_2} \frac{c_3}{c_2 + c_3}}$$

例 3. 传染病模型

随着卫生设施的改善、医疗水平的提高以及人类文明的不断发展,诸如霍乱、天花等曾经肆虐全球的传染性疾病已经得到有效的控制.但是一些新的、不断变异着的传染病毒却悄悄向人类袭来. 20 世纪 80 年代十分险恶的爱滋病毒开始肆虐全球,至今仍在蔓延; 2003 年春来历不明的 SARS 病毒突袭人间,给人们的生命财产带来极大的危害.长期以来,建立传染病的数学模型来描述传染病的传播过程,分析受感染人数的变化规律,探索制止传染病蔓延的手段等,一直是各国有关专家和官员关注的课题.试建立传染病的数学模型,以分析传染病的传播规律.

模型1 已感染人数(病人)

假设每个病人每天有效接触(足以使人致病)人数为 λ, 建立如下数学模型

$$i(t + \Delta t) - i(t) = \lambda i(t) \Delta t$$

若有效接触的是病人,则不能使病人数增加,必须区分已感染者(病人)和未感染者(健康人).

模型 2 区分已感染者(病人)和未感染者(健康人)(SI 模型)

假设:

1)总人数 N 不变, 病人和健康人的比例分别为 i(t), s(t).

2)每个病人每天有效接触人数为 λ, 且使接触的健康人致病, λ 为日接触率 建立数学模型

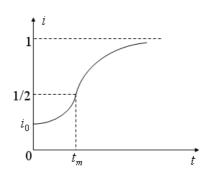
$$N[i(t + \Delta t) - i(t)] = [\lambda Ns(t)]i(t)\Delta t$$

由
$$\frac{di}{dt} = \lambda si$$
, $s(t) + i(t) = 1$ 得

$$\begin{cases} \frac{di}{dt} = \lambda i (1 - i) \\ i(0) = i_0 \end{cases}$$

称为 Logistic 模型. 可解出

$$i(t) = \frac{1}{1 + \left(\frac{1}{i_0} - 1\right)}e^{-\lambda t}$$



$$\diamondsuit t_m = \lambda^{-1} \ln \left(\frac{1}{i_0} - 1 \right), t_m$$
 是传染病高潮到来时刻. λ (日接触率) $\longrightarrow t_m \uparrow$.

 $t \to \infty \Rightarrow i \to 1$, 说明病人可以治愈!

模型 3 传染病无免疫性——病人治愈成为健康人,健康人可再次被感染(SIS 模型)

增加假设

3)病人每天治愈的比例为μ,即日治愈率.

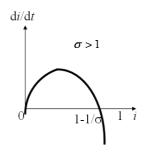
建立数学模型

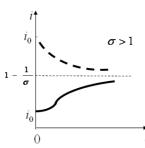
$$N[i(t + \Delta t) - i(t)] = \lambda Ns(t)i(t)\Delta t - \mu Ni(t)\Delta t$$

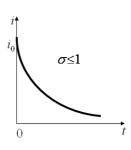
$$\begin{cases} \frac{di}{dt} = \lambda i(1-i) - \mu i \\ i(0) = i_0 \end{cases}$$

 λ 是日接触率, $1/\mu$ 是感染期 $\sigma=\lambda/\mu$ 是一个感染期内每个病人的有效接触人数, 称为接触数.

$$\frac{di}{dt} = -\lambda i [i - (1 - \frac{1}{\sigma})], \quad i(\infty) = \begin{cases} 1 - \frac{1}{\sigma}, & \sigma > 1 \\ 0, & \sigma \le 1 \end{cases}$$







感染期内有效接触感染的健康者人数不超过病人数.

思考

模型 2(SI 模型)如何看作模型 3(SIS 模型)的特例

模型 4 传染病有免疫性——病人治愈后即移出感染系统, 称移出者 SIR 模型 假设

1)总人数 N 不变、病人、健康人和移出者的比例分别为 i(t), s(t), r(t)

2)病人的日接触率 λ , 日治愈率 μ , 接触数 $s=\lambda/\mu$.

建立数学模型

$$s(t) + i(t) + r(t) = 1$$

需建立 i(t), s(t), r(t)的两个方程

$$N[i(t + \Delta t) - i(t)] = \lambda Ns(t)i(t)\Delta t - \mu Ni(t)\Delta t$$

$$N[s(t + \Delta t) - s(t)] = -\lambda Ns(t)i(t)\Delta t$$

$$\begin{cases} \frac{di}{dt} = \lambda si - \mu i \\ \frac{ds}{dt} = -\lambda si \\ i(0) = i_0, s(0) = s_0 \end{cases}$$

 $i_0+s_0\approx 1$ (通常 $r(0)=r_0$ 很小)

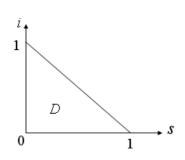
无法求出 i(t), s(t)的解析解,在相平面 s-i 上研究解的性质. 由上模型得

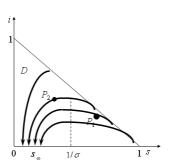
$$\begin{cases} \frac{di}{ds} = \frac{1}{\sigma S} - 1\\ i \Big|_{s=s_0} = i_0 \end{cases}$$

相轨线

$$i(s) = (s_0 + i_0) - s + \frac{1}{\sigma} \ln \frac{s}{s_0}$$

定义域 $D = \{(s,i) | s \ge 0, i \ge 0, s+i \le 1\}$, 在 D 内作相轨线 i(s)的图形, 进行分析





i(s)图形: $s(t) \downarrow$, $i_{\infty} = 0$, $i(s = 1/\sigma) = i_m$, s_{∞} 满足

$$s_0 + i_0 - s_\infty + \frac{1}{\sigma} \ln \frac{s_\infty}{s_0} = 0$$
.

 $s_0 > 1/\sigma(P_1) \rightarrow i(t)$ 先升后降至 0, 传染病蔓延.

 $s_0 < 1/\sigma(P_2) \rightarrow i(t)$ 单调降至 0, 传染病不蔓延.

预防传染病蔓延的手段

传染病不蔓延的条件—— $s_0 < 1/\sigma$.

λ(日接触率)↓⇒ 卫生水平↑.

μ(日治愈率)↑⇒ 医疗水平↑.

降低 s_0 $(s_0 + i_0 + r_0 = 1) \Rightarrow r_0 \uparrow$, 即群体免疫.

 σ 的估计: 由 $s_0+i_0-s_\infty+rac{1}{\sigma}\lnrac{s_\infty}{s_0}=0$,忽略 i_0 ,得 $\sigma=rac{\ln s_0-\ln s_\infty}{s_0-s_\infty}$.

第四章 图论模型

图论最早起源于一些数学游戏的难题研究,如欧拉所解决的哥尼斯堡七桥问题,以及在民间广泛流传的一些游戏难题,如迷宫问题、博奕问题、棋盘上马的行走路线问题等.这些古老的难题,当时吸引了很多学者的注意.在这些问题研究的基础上又继续提出了著名的四色猜想和汉米尔顿(环游世界)数学难题.

1847 年,图论应用于分析电路网络,这是它最早应用于工程科学,以后随着科学的发展,图论在解决运筹学,网络理论,信息论,控制论,博奕论以及计算机科学等各个领域的问题时,发挥出越来越大的作用.在人们的社会实践中,图论已成为解决自然科学、工程技术、社会科学、生物技术以及经济、军事等领域中许多问题的有力工具之一,因此越来越受到数学家和实际工作者的喜爱.我们所学的这一章只是介绍一些基本概念、原理以及一些典型的应用实例,目的是在今后对工程技术有关学科的学习研究时,可以把图论的基本知识、方法作为工具.

第一节 图论的基本概念

一、图的定义

图 G 是指由非空有限集合 V(G) 和 V(G)中某些元素的无序对的集合 E(G)构成的二员组(V(G), E(G)), 其中 V(G) 成为 G 的顶点集合,其中的元素成为 G 的顶点. E(G) 成为 G 的边集,其中的元素称为 G 的边. 一般简单记为 G=(V(G), E(G)),在不发生混淆的情况下,可记为 G=(V, E). 如果 $V=\{v_1, ..., v_n\}$,那么 E 中的元素 e 与 V 中某两个元素构成的无序对 $\{v_i, v_j\}$ 相对应,记为 $e=v_iv_j$ 或 $e=v_jv_i$. 图可用图形来表表示,如:



简单图: 既没有环也没有重边的图. 开始我们都是讲简单图.

二、点与点,点与边,边与边间的关系

若 $e=v_iv_i\in E$, 则称顶点是 v_i 和 v_i 相邻, 并称 v_i , v_i 为边 e 的端点, 也称 e 与 v_i , v_i 关联.

若 e_1, e_2 ∈ E, 且 e_1 和 e_2 有公共的端点,则称 e_1 与 e_2 是相邻的.

三、阶与度

- 1、图的阶:一个图的顶点数.
- 2、顶点的度 $\mathbf{d}_G(v)$: 图 G 中和 V 关联的边的数目(与 V 关联的每个环算作两条边). 顶点有偶点和 奇点之分. 当 $\mathbf{d}_G(v)$ =0 时称其为孤立点.

结论 1: 设 G=(V,E)是一个图,则有 $\sum_{v\in V}d_G(v)=2|E|$.即奇点的个数必为偶数.

四、点边序列

若序列: $v_0e_1v_1...e_kv_k$, 其中 $e_i \in E$, $v_j \in V$, i=1,2,...,k, j=0,1,...,k, 且相邻两项有关联关系, 则称此序列是一条道路(途径、通道), 其中称 v_0 为起点、称 v_k 为终点、而 k 称为此道路的长.

- 1、迹: 如果 G 中的一条道路 W 上的边互不相同, 则称 W 为 G 的迹.
- 2、链: 如果 G 中的一条道路 W 上的点互不相同, 则称 W 为 G 的链(路), 若 W 的起点、终点分

别为 v_i, v_j , 则 W 可简记为 (v_i, v_j) 或 $P(v_i, v_j)$.

- 3、闭通道: 如果通道的长至少为1,且起点和终点重合,则称该通道为闭通道.
- 4、闭迹: 起点和终点重合且长至少1的迹称为闭迹.
- 5、圈: 起点和内部顶点互不相同的闭迹称为圈.

五、子图

- 1、子图: 如果图 G 和图 H 满足 $V(H) \subset V(G)$, $E(H) \subset E(G)$, 则称 H 是 G 的子图, 记为 $H \subset G$.
- 2、支撑子图: 如果 V(H)=V(G), 且 $E(H) \subseteq E(G)$, 则称 H 为 G 的支撑子图.
- 3、导出子图: 设 V 是图 G 的顶点集 V(G)的一个非空子集、令

$$E' = \{e' \mid e' \in E(G), e' = v_i v_i, v_i, v_i \in V'\}$$

则称 G 的子图 G'=(V',E')为由 V 导出的子图,记为 G'=G[V'].

设 $E' \subseteq E(G)$, $E' \neq \emptyset \diamondsuit V' = \{v | v \notin E'$ 中某条边的端点}

则称 G 的子图 G' = (V', E') 为由 E' 导出的子图,记为 G' = G[E'].

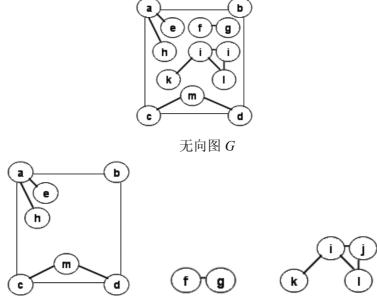
4、生成子图: 设 G 与 H 是两个图, 且 V(H) = V(G), $E(H) \subset E(G)$, 则称 H 为 G 的生成子图.

六、图的连通性

1、连通图: 如果对于图 G 的任意两个顶点 v_i 和 v_j , G 中都存在链 $P(v_i,v_j)$, 则称图 G 是连通图. 否则称为非连通图.

- 2、连通子图: 如果 $H \subset G$, 且H是连通图, 则称H是G的连通子图.
- 3、极大连通子图: 指 H 为 G 的连通子图, 并且 G 中不存在连通子图 H' 使得 $H \subseteq H'$, $H \neq H'$. 此时也称它为 G 的连通分支. 从而可知, 连通图恰有一个连通分支, 而非连通图则有两个或两个以上的连通分支.

无向图的连通性的图例

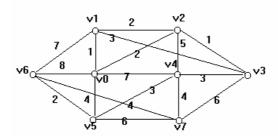


无向图 G 的三个连通分量

七、连通图的应用

8 个城市 v_0 , v_1 , ..., v_7 之间有一个公路网(如图所示), 每条公路为图中的边, 边上的权数表示

通过该公路所需的时间. 设你处在城市 v_0 , 那么从 v_0 到其他各城市, 应选择什么路径使所需的时间 最短?



本例比较简单,可以用穷举法选择出从 ν_0 到其他各城市之间的最短路,即运行时间最短的路. 然而当公路网比较复杂时,是否有好的算法呢?

狄克斯特拉在1959年提出一种最短路的算法,至今仍被认为是最好的算法之一.

设 $(v_0, u_1, u_2, ..., u_n)$ 为从 v_0 到 u_n 的最短路,记

$$d(v_0, u_n) = \omega(v_0, u_1) + \omega(u_1, u_2) + \dots + \omega(u_{n-1}, u_n),$$

称之为从 v_0 到 u_n 的最短路的权数. 若S为V的任意子集,则称 $d(v_0,S) = \min_{u \in S} d(v_0,u)$ 为从 v_0 到S的最短路的权数.

对于例 1,设 S 为节点集 V 的一个节点子集, $v_0 \in S$,设 $S^c = V \setminus S$ 为 S 的节点余集.容易看出, 若($v_0, u_1, u_2, ..., u, v$) 为从 v_0 到 S^c 的最短路,则必有 $u \in S, v \in S^c$,使($v_0, u_1, u_2, ..., u$) 为从 v_0 到 u 的最短路.于是

$$\begin{cases} d(v_0, v) = d(v_0, S^c) \\ d(v_0, v) = d(v_0, u) + \omega(u, v) \\ d(v_0, S^c) = \min_{\substack{u \in S \\ v \in S^c}} \{d(v_0, u) + \omega(u, v)\} \end{cases}$$

上式为狄克斯特拉算法的理论基础. 下面以例 1 为例说明算法步骤:

设
$$S_0 = \{v_0\}, S_0^c = \{v_1, v_2, \dots, v_7\}, 求$$

$$d(v_0, S_0^c) = \min_{u \in S_0} \left\{ d(v_0, u) + \omega(u, v) \right\} = \min_{v \in S_0^c} \{ d(v_0, v) + \omega(u, v) \}$$

容易看出, $d(v_0, S_0^c) = \omega(v_0, v_1) = 1$.

把 v_1 加入 S_0 ,设 $S_1 = \{v_0, v_1\}$, $S_1^c = \{v_2, v_3, \dots, v_7\}$,再求

$$d(v_0, S_1^c) = \min_{u \in S_1} \{d(v_0, u) + \omega(u, v)\} = d(v_0, v_2) = 2.$$

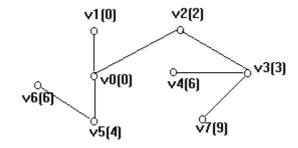
$$v \in S_1^c$$

把 v_2 加入 S_1 , 设 $S_2 = \{v_0, v_1, v_2\}$, $S_2^c = \{v_3, v_4, \dots, v_7\}$, 求

$$d(v_0, S_2^c) = \min_{u \in S_2} \{d(v_0, u) + \omega(u, v)\} = d(v_0, v_2) + \omega(v_2, v_3) = d(v_0, v_3) = 3$$

$$v \in S_2^c$$

以此类推,可求出从 v_0 到其它任意节点的最短路和它们的权数. 若把从 v_0 到 $v_j(j=1,2,...,7)$ 的最短路的权数标在该点处,并令 v_0 处的权数为零,则所得结果如图所示.



实际上, 它是原图的以 v₀ 为根的生成树, 直接表明从 v₀ 到其它各节点的最短路径.

八、图的运算

1、减: 若 $E_1 \subset E(G)$, 从图 G 中删去 E_1 的所有边得到得图, 记为 G- E_1 .

若 $V_1 \subseteq V(G)$,且 $V_1 \neq V(G)$,则把从图 G 中删去 V_1 得所有项点以及 V_1 中项点关联得边得到得图记为 $G-V_1$. 易得 $G[V']=G-(V(G)\setminus V')$.

- 2、加: 若 $H \subseteq G$, $E_2 \subseteq E(G) \setminus E(H)$, 且 E_2 中每条边得端点都属于 V(G), 在图 H 中添上 E_2 得所有边得到得图记为 $H+E_2$.
- 3、并: 若 $H_1 \subseteq G$, $H_2 \subseteq G$, 则把顶点集为 $V(H_1) \cup V(H_2)$, 边集为 $E(H_1) \cup E(H_2)$ 的图成为 H_1 和 H_2 的并, 记为 $H_1 \cup H_2$. 若 $E(H_1) \cap E(H_2) = \emptyset$, 则称 $H_1 \cup H_2$ 为 H_1 与 H_2 的和, 记为 $H_1 + H_2$.

九、边割

设 G=(V,E)是一个图, S 为 V 的一个非空真子集, $\overline{S}=V\setminus S$, 记

$$[S, \overline{S}] = \{v_i v_j \in E \mid v_i \in S, v_j \in \overline{S}\}$$

如果 $[S,\overline{S}]\neq\emptyset$,则称 $[S,\overline{S}]$ 是G的一个边割。若E'是G的一个边割,但E'的任何真子集都不是G的边割,则称E为G的极小边割,此时称之为G的割集。

结论 1: 设 C 和 $[S,\overline{S}]$ 分别时图 G 的圈和边割,则

$$|E(C)\cap[S,\overline{S}]|\equiv 0 \pmod{2}$$

十、有向图的基本概念

1、有向图 D: 指由非空有限集合 V(D) 和 V(D)中某些元素的无序对的集合 A(D)构成的二员组 (V(D), A(D)), 其中 V(D)成为 D 的顶点集合, 其中的元素成为 D 的顶点. A(D)成为 D 的弧集, 其中的元素称为 D 的弧. 在不发生混淆的情况下,可记为 D(V,A).

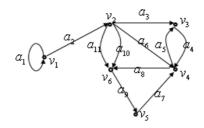
如果 $V(D) = \{v_1, v_2, \dots v_n\}$,那么A中的元素a与V(D)中某两个元素构成的有序对 $\{v_i, v_j\}$ 相对应,记为 $a = \{v_i, v_i\}$. 其中 v_i 成为a的尾, v_i 称为a的头,a称为 v_i 的出弧,也称为 v_i 的入弧.

顶点 ν 的在 D 中的出弧的数目记为 $d_D^+(D)$, 称之为 ν 的出度;

顶点 v 在 D 中入弧的数目记为 $d_p(D)$, 称之为 v 的入度.

结论 3: 设
$$D=(V,A)$$
是一个有向图,则 $\sum_{v\in V}d_D^+(v) = \sum_{v\in V}d_D^-(v) = |A|$

注意: 在无向图中的定义(例如通道、迹等)也可相应的扩充为有向图的情况.



十一、几类重要的图

1、完全图: 任何一对顶点都相邻的简单图. N 阶完全图记为 K_n . 划分: 设 S 是一个集合, 如果存在 k 个集合 $S_1, S_2, ..., S_k$, 使得

$$S = \bigcup_{i=1}^k S_i \coprod S_i \cap S_j = \emptyset (i \neq j)$$

则称 $S_1, S_2, ..., S_k$ 为 S 的一个划分.

2、二部图(二分图): 设 G 是一个图, 如果存在 V(G)的一个划分 X,Y, 使得 G 的任何一条边的一个端点在 X 中, 另一个在 Y 中, 则称 G 为二部图, 记作 G=(X,Y,E).

完全二部图是指这样的简单二部图 G=(X,Y,E),其中 X 的每个顶点都与 Y 的每个顶点相邻. 若 |X|=m,|Y|=n,则完全二部图 G=(X,Y,E)记 为 K_{mn} .

结论 4: 图 G 是二部图当且仅当 G 中不包含奇圈.

3、Euler 图:是指图 G 中存在包含一切边的闭迹 W.

结论 5: 图 G 是 Euler 图当且仅当 G 中无奇点.

4、Hamlton 图: 是指图 G 中存在一条包含所有顶点的圈 C.

结论 6: G 是一个(p,q)图, $p \ge 3$. 若 $\forall u, v \in V$, 有 $d_G(u) + d_G(v) \ge p$,则称图 G 是 Hamlton 图.

5、无向网络(赋权图): 在边加权w的图 G.其中w是 E(G)上的一个实值函数, 称为G的权函数. 这时 G 记为 G=(V(G),E(G),w).

第二节 图的存储结构

邻接矩阵和加权邻接矩阵(labeled adjacency matrix)

•无权值的有向图的邻接矩阵

设有向图具有 n 个结点,则用 n 行 n 列的布尔矩阵 A 表示该有向图;

并且 A[i,j]=1,如果 $i \subseteq j$ 有一条有向边; A[i,j]=0 如果 $i \subseteq j$ 没有一条有向边注意: A[i,i]=0. 出度:i 行之和. 入度:j 列之和.



•无权值的无向图的邻接矩阵

设无向图具有 n 个结点,则用 n 行 n 列的布尔矩阵 A 表示该无向图;并且 A[i,j]=1,如果 i 至 j 有一条无向边;A[i,j]=0 如果 i 至 j 没有一条无向边注意:A[i,i]=0. i 结点的度:i 行或 i 列之和.上三角矩阵或下三角矩阵.



•有向图的加权邻接矩阵

设有向图具有n个结点,则用n行n列的矩阵A表示该有向图;

并且 A[i,j]=a,如果 $i \subseteq j$ 有一条有向边且它的权值为 a. A[i,j]=空或其它标志; 如果 $i \subseteq j$ 没有一条有向边.



优点: 判断任意两点之间是否有边方便, 仅耗费 O(1)时间.

缺点: 即使 $<< n^2$ 条边, 也需内存 $n^2/2$ 单元, 太多;仅读入数据耗费 $O(n^2)$ 时间, 太长.

第三节 树与图的生成树

若一个连通图中不存在任何回路,则称为树(如图 3-1 所示). 由树的定义,直接得下列性质:

- (1)树中任意两节点之间至多只有一条边:
- (2)树中边数比节点数少1;
- (3)树中任意去掉一条边, 就变为不连通图;
- (4)树中任意添加一条边, 就会构成一个回路.

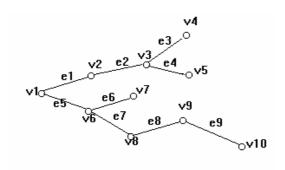


图 3-1

最小生成树及其算法

不难看出,任意一个连通图或者就是一个树,或者去掉一些边后形成一个树.一个连通图去掉一些边后形成的树称之为该连通图的生成树.

一般来说,一个连通图的生成树可能不止一个. 设 T 为图 G 的一个生成树,若把 T 中各边的权数相加所得的和数称之为生成树 T 的权数. G 的所有生成树中,权数最小者称为 G 的最小生成树.

求最小生成树问题有很广泛的实际应用. 例如把n个城市用高压电缆连接起来建立一个电网. 如何能设计一个把n个城市联系起来的电网, 使所用的电缆长度之和最短, 即费用最小(因费用与长度正比), 就是一个求最小生成树的问题.

由树的性质及生成树的定义知,树T为图G的最小生成树的充分必要条件是对T以外的任意边 (v_i,v_i) ,

$$\omega(v_i, v_j) \ge \max\{\omega(v_i, v_{i_1}), \omega(v_{i_1}, v_{i_2}), \dots, \omega(v_{i_k}, v_{j})\}$$

其中 $(v_i, v_{i_i}, v_{i_2}, \dots, v_{i_k}, v_j)$ 为生成树 T(V, E)的连接 v_i 和 v_j 的路,故 G 最小生成树 T 必然由那些权数较小而不形成任何回路的边组成.下面所介绍的算法都是根据这个基本原理得到的.

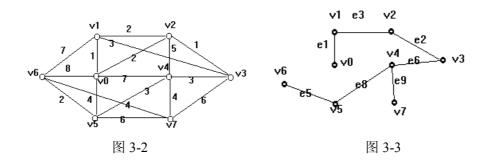
1. 克罗斯克尔算法

克罗斯克尔算法是1956年提出的,俗称"避圈法".步骤如下:

设G为由n个节点组成的连通赋权图.

- (1)先把G中所有边按权数大小由小到大重新排列,并取权数最小的一条边作为T的一条边.
- (2)从剩下的边中按(1)的排列顺序取下一条边,若该边与前已取进T中的边没有形成回路,则把该边取入T中作为T的一条边,否则舍去,继续按(1)的排列顺序再取下一条边,…,如此下去直至有n-1条边组成G的最小生成树T.

以图 3-2 为例(节点数 n=8). 先按各边权数的权数由小到大排列 $e_1(v_0,v_1)$, $e_2(v_2,v_3)$, $e_3(v_1,v_2)$, $e_4(v_0,v_2)$, $e_5(v_5,v_6)$, $e_6(v_3,v_4)$, $e_7(v_1,v_2)$, $e_8(v_4,v_5)$, $e_9(v_4,v_7)$, $e_{10}(v_0,v_5)$,



由克罗斯克尔算法的步骤, 顺次取边 e_1 , e_2 , e_3 , e_4 , e_5 , e_6 , e_7 , e_8 , e_9 , 舍去 e_4 和 e_7 , 得到最小生成树为 $T=\{e_1,e_2,e_3,e_5,e_6,e_8,e_9\}$, 即为图 3-3 所示.

为了便于编程, 我们采用所谓"最小标号法"对节点进行重新编号, 方法如下:

先对图中各节点以自然方式编号,如

以后每一步对取进T中边的节点都要重新编号: 若 $(u_1, u_2, ..., u_r)$ 为T中任意一条路,则该路所经过的节点 $u_1, u_2, ..., u_r$ 都要重新标以它们中最小的标号,即

$$l(u_1) = l(u_2) = \cdots = l(u_r) = \min\{u_1, u_2, \cdots, u_r\}$$
.

经过这种"最小标号法"重新编号后,就能判断下一条边 $e_k(v_{k_i},v_{k_j})$ 是否与前已取进T中的边构成某回路,因此决定 $e_k(v_{k_i},v_{k_j})$ 是否能取进T中。若 $l(v_{k_i})=l(v_{k_j})$,则说明 $e_k(v_{k_i},v_{k_j})$ 取进T中会形成一个回路,应舍去 $e_k(v_{k_i},v_{k_j})$;若 $l(v_{k_i})=l(v_{k_j})$,说明至少有一个节点不在前已取进的T的节点中,故在T中加入 e_k 后不会形成任何回路,因而应将 $e_k(v_{k_i},v_{k_j})$ 取进T中。如此下去,最终能得到最小生成树T.

克罗斯克尔算法的流程图与程序请读者自行完成.

2. 普莱姆算法

普莱姆算法的基本思想为:

- (1)先在 G 中任取一个节点 ν_0 , 并取入 T 中;
- (2)令 $S=V(G)\setminus V(T)$, 其中 V(G), V(T)分别为 G 与 T 的节点集;
- (3)在所有连接 V(T)的节点与 S 的节点的边中, 选出权数最小的边(u_0,v_0)即

$$\omega(u_0, v_0) = \min \{ \omega(u, v) \mid u \in V(T), v \in S \}$$

(4)将边 (u_0,v_0) 取入T中. 重复(2)至(4)步骤, 直至G中的节点全都取入T中为止.

仍以图 3-2 为例,若 v_0 为起点,顺次取入 T 中的边为(v_0,v_1),(v_2,v_3),(v_3,v_4),(v_4,v_5),(v_5,v_6),(v_4,v_7)或(v_6,v_7)形成的最小生成树 T,与图 3-3 一致.

类似可定义最大生成树及其相应的算法.

第四节 网络流及其应用

在我们的日常生活中有大量的网络,如电网、水管网、交通运输网、通讯网、生产管理网等.近二、三十年来在解决网络方面的有关问题时,网络流理论及其应用起着很大的作用.

先看一个实例.设有一个水管网络,该网络只有一个进水口和一个出水口,其它管道(边)和接口(节点)均密封.网络中每个管道用它的截面面积作为该管道的权数,它们反映管道在单位时间内可能

通过的最大量称为容量.现在在此水管道网络中注入稳定的水流,水由进水口注入经过水管网络后流向出水口,最后从出水口流出,这就形成一个实际的稳定流动,称为流.分析这种实际流动,有如下性质:

- (1)实际流动是一个有向的流动;
- (2)每个管道中单位时间内通过的流量不可能超过该管道的容量(权数);
- (3)每个内部节点处流入节点的流量与流出节点的流量应相等;
- (4)流入进水口的流量应等于流出出水口的流量,即为实际流动的流量.

如果我们进一步加大流量,由于受水管网络的限制,加到一定的流量后,再也加不进去了,这就是此水管网络能通过的最大流量.这里所谓的网络流理论正是从这些实际问题中提炼出来的,下面来介绍有向图的网络流理论及其应用.

网络流与最大流最小截集定理

网络流理论最初由福特和弗克森于 1956 年创立的,包括理论与算法两部分.

网络流的基本问题如下:设一个有向赋权图 G(V, E), $V=\{s,a,b,c,...s'\}$,其中有两个特殊的节点 s 和 s'. s 称为发点,s'称为收点.图中各边的方向和权数表示允许的流向和最大可能的流量(容量),并假设容量均为整数.问在这个网络图中从发点流出到收点汇集,最大可通过的实际流量为多少?流向分布情况怎样?

网络流理论的关健是引进一个"流"的概念.

在有向赋权图 G(V, E)的边集 E 上定义一个整数值函数 f(x,y), (x,y)为 E 中的有向边, 记 V(x)为与节点 x 邻接的节点的集合, 使得满足条件:

- (1)f(x,y)=-f(y,x);
- $(2)f(x,y) \leq \omega(x,y);$
- (3) $\Sigma f(x,y)=0$,当 $x\neq s$, s'时,其中 $y\in V(x)$;
- $(4)f(s,y) \ge 0, f(y',s') \ge 0, y \in V(s), y' \in V(s').$

则称 f(x,y)为有向赋权图 G(V,E)上的一个流.

条件(1)表示从x到y的正流等于从x到y的负流;条件(2)表示每条边的实际流量不能超过该边的容量;条件(3)表示每个内部节点(即不是发点或收点的节点)的流量之和必为零,即满足零守恒定律——流进内节点的流量之和等于流出该节点的流量之和;条件(4)表示发点处只能流出,收点处只能流进.

设f为图G(V,E)上的一个流,则

$$Q(f) = \sum_{x \in V(s)} f(s, x)$$

称为f的流值,它是流f在发点处的总流出量.以后还将看到,Q(f)也等于流f在收点的总流入量,即

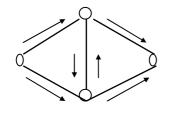
$$Q(f) = \sum_{y' \in V(s')} f(y', s')$$

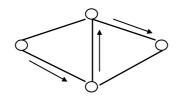
在展开网络流理论之前, 先来看一个例子.

例 1 有向赋权图如图 4-1 所示. 图中未标出的有向边(如 (a,s),(b,s),(s',a),(s',b))都假定为零.

首先注意到,ƒ≡0总是一个流,称为零流.

图 4-2 和图 4-3 表示另外两个流. 其中未标出的边总假定为零. 由条件(1)知, 反方向边的流必为正方向边的流的负值.





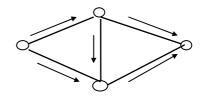


图 4-1

图 4-2

图 4-3

 $Q(f_1)=1$, $Q(f_2)=4$. 本例表明,一个已知的图可以有不同的流和流值.那么,如何来判断和寻找具有最大流值的流呢?为此,我们还要引入另一个概念——截集.

设 C 为包含发点但不包含收点的节点子集,C' 为 C 的余集(C'=VC),C' 显然,包含收点而不包含发点,称节点集 V 的这种分解(C, C')为网络 G 的一对截集. 从 C 到 C' 的各边的权数之和称为截集(C, C')的容量,记为

$$\omega$$
 (C, C') = $\sum_{\substack{x \in C \\ y \in C'}} \omega(x, y)$.

在例 1 中, 设 $C = \{s,a\}, C' = \{b,s'\}$ 则 $\omega(C,C') = \omega(s,b) + \omega(a,b) + \omega(a,s') = 4$.

设
$$f$$
为 G 的一个流,则显然有 $f(C,C') \le \omega(C,C')$,其中 $f(C,C') = \sum_{\substack{x \in C \\ v \in C'}} f(x,y)$.

对于任意给定的图 G, G 上不同的截集有不同的容量. 同时 G 上不同的流有不同的流值. 称具有最小容量的截集为最小截集, 具有最大流值的流称为最大流. 网络流理论的基本定理将证明最大流的流值等于最小截集的容量. 先看一个引理:

引理 1 若 f 为图 G(V, E)的一个流,则对任何截集(C, C')有 O(f) = f(C, C').

证明 则流的条件(1), 可知 f(C,C)=0. 故

$$f(C,C') = f(C,C') + f(C,C) = f(C,V)$$

但是
$$f(C,V) = \sum_{x \in C} f(x,V)$$
. 当 $x \neq s$ 时, $f(x,V) = f(x,V(x)) = \sum_{y \in V(x)} f(x,y) = 0$. 于是

$$f(C,V) = \sum_{x \in C} f(x,V) = f(s,V)$$

即有

$$Q(f) = f(s,V) = f(C,C')$$
.

定理 3 (最大流最小截集定理)设 G(V, E)为赋整数权的有向图,则 f_0 为 G 的最大流, (C_0, C_0') 为 G 的最小截集的充分必要条件是

$$Q(f_0) = f_0(C_0, C'_0) = \omega(C_0, C'_0)$$
.

证明: 设 f(x,y)为整数值,由于 Q(f)被任一对截集的容量所限,故最大流必存在.设 f_0 为 G 的一个最大流,我们选取节点子集 C_0 如下:

设 $f_0(x,y) < \omega(x,y)$,称边(x,y)为 f_0 的非饱和边. 若一条路的每一条边都是非饱和的,则称该路为 f_0 的非饱和路;若存在从 S 出发的非饱和路达到的节点,称为 f_0 的非饱和节点. C_0 是所有 f_0 的非饱和节点组成的集合.

我们断言 C_0 不包含收点 s' . 因为如果 $s' \in C_0$,则从 s 到 s' 必存在一条非饱和路 R . 沿着这条非饱和路,在该路的每条边上把 f_0 的值都增加 1,而得到一个新流 f_1

$$f_1(x,y) = \begin{cases} f_0(x,y) + 1, & (x,y) \in R \\ f_0(x,y), & (x,y) \notin R \end{cases}$$

显然 $Q(f_1)=Q(f_1)+1$, 这与 f_0 为最大流的假设矛盾.

因 C_0 包含 s 而不包含 s',故 C_0 与它的余集 C_0' 构成一对截集. 由 C_0 的定义可知,对任意 $x \in C_0, y \in C_0'$,有 $f_0(x,y) < \omega(x,y)$. 因而

$$f_0(C_0, C_0') = \omega(C_0, C_0') = Q(f_0)$$
(3)

又由式(2)和(3)得知,对任意的一对截集 (C_1) ,有

$$\omega(C,C') \ge f_0(C,C') = f_0(C_0,C'_0) = \omega(C_0,C'_0)$$
.

因此式(3)表明, 最大流的流值等于最小截集的容量.

定理3得到了判断最大流的准则,同时也提供了一个构造最大流的算法.

最大流的算法

设给定一个有向赋权图 G(V, E), 从任一可行流出发, 例如从零流 $f_1(x,y) \equiv 0$ 出发. 令 C_1 为由发点 S_1 和对于 S_2 的非饱和点组成的节点子集.

若收点 $s' \notin C_1$,则 f_1 为最大流;

若收点 $s' \in C_1$,则 f_1 不是最大流,从而可由下列方法加以改进.因 $s' \in C_1$,故存在从 s 到 s' 的非饱和路 R,在 R 上的每边 $(x,y) \in R$,均有 $f_1(x,y) < \omega(x,y)$, $(x,y) \in R$.

令 $a = \min \left\{ \min_{(x,y) \in \mathbf{R}} \{ \omega(x,y) - f_1(x,y) \}, \min_{-(x,y)} \{ -f_1(x,y) \} \right\}$,其中, $-(x,y) \in R$ 表示边(x,y)的方向与路 R的方向相反. 并令

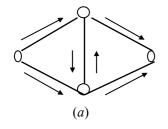
$$f_2(x,y) = \begin{cases} f_1(x,y) + a, & (x,y) \in R \\ -f_1(x,y) - a, & -(x,y) \in R \\ f_1(x,y), & 其它边 \end{cases}$$

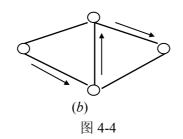
容易验证, f_2 也为可行流, 且它的流值为 $Q(f_2)=Q(f_1)+a$, $a\geq 1$.

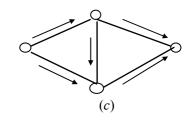
再以 f_2 为基础, 重复上面的步骤, 必可经过有限步后得到一个最大流. 这个算法称为非饱和算法.

以例 1 为例,令 f_1 \equiv 0,则 C_1 = $\{s,a,b,s'\}$,任选一条从 s 到 s' 的非饱和路,如 R_1 = $\{s,a,b,s'\}$, a_1 =min $\{5,2,4\}$ =2. 令

$$f_2(x,y) = \begin{cases} 2, & (x,y) = (s,a), (a,b), (b,s') \\ 0, & (x',y) = (s,b), (a,s') \end{cases}$$







反向边均取负值,如图 4-4(a)所示.

同理可得关于 f_2 的非饱和点集 $C_2\{s,a,b,s'\}$. 再选一条从 s 到 s' 的非饱和路, 如 R_2 =(s,a,s'), a_2 =min{5-2,1}=1. 令

$$f_3(x,y) = \begin{cases} f_2(x,y) + 1, & (x,y) \in R_2 \\ f_2(x,y), & \text{!`E'} \end{cases}$$

 f_3 如图 4-4(b)所示. 类似地,得 C_3 = {s,a,b,s'},但从 s 到 s' 的非饱和路只有一条 R_3 =(s,b,s'). a_3 =min{1-0, 4-2,}=1. 令

$$f_4(x,y) = \begin{cases} f_3(x,y) + 1, & (x,y) \in R_3 \\ f_3(x,y), & 其它 \end{cases}$$

 f_4 如图 4-4(c)所示. 这时 C_4 = $\{s,a\}$, s'不包含在 C_4 中因而 f_4 为 G 的一个最大流, 它的流值为 $Q(f_4)$ =4. 在编程时也可采用标号法, 并选用迭代法来实现.

网络流的应用

1. 最小费用最大流问题

在实际问题中,人们不仅考虑流量的大小,还要考虑输送这些流量所需的费用、代价等.例如某工厂要把产品运往销售点,在选择运输路线时,不仅要考虑运输量,而且还要考虑运输最大运输量时的最小运费问题,这是典型的最小费用最大流问题之一.

设一个网络 G(V, E), E 中每条边(i,j)对应一个容量 c(i,j)(如最大输送能力),以及一个 a(i,j)表示输送单位流量所需的费用. 如果有一个运输方案(可行流),流量为 f(i,j),则最小费用最大流问题就是求下列的极值问题

$$\min_{f \in F} a(f) = \min_{f \in F} \sum_{(i,j) \in E} a(i,j)f(i,j)$$

其中F为G的最大流的集合,即在最大流中寻找一个费用最小的最大流.

确定最小费用流的基本思想是从零流为初始可行流开始,在每次迭代过程中对每条边赋予与c(i,j)(容量)、a(i,j)(单位流量运输费用)、f(i,j)(现有流的流量)有关的权数 $\omega(i,j)$,以形成一个有向的赋权图. 再求最短路的方法确定由 s 到 s' 的费用最小的非饱和路,沿着该路增加流量,得到相应的新流. 经过多次迭代,直至达到最大流为止.

构造权数的方法如下: 对任意边(i,j),根据现有的流 f,该边上的流量可能增加(f(i,j) < c(i,j)),也可能减小(f(i,j) > 0),因此每条边赋予向前费用权 $\omega^+(i,j)$ 与向后费用权 $\omega^-(i,j)$:

$$\omega^{-}(i,j) = \begin{cases} -a(i,j), & \quad \text{若} f(i,j) > 0 \\ +\infty, & \quad \text{若} f(i,j) = 0 \end{cases}$$

对赋权后的有向图, 如果把权 $\omega(i,j)$ 看作长度, 即可确定从 s 到 s' 的费用最小的非饱和路, 它等价于确定从 s 到 s' 的最短路. 确定了非饱和路后就可确定该路的最大可增流量, 因此需对每一条边确定

一个向前可增流量 $\Delta^+(i,j)$ 与向后可增流量 $\Delta^-(i,j)$:

$$\Delta^{+}(i,j) = \begin{cases} c(i,j) - f(i,j), & \exists f(i,j) < c(i,j) \\ 0, & \exists f(i,j) = c(i,j) \end{cases}$$
$$\Delta^{-}(i,j) = \begin{cases} f(i,j), & \exists f(i,j) > 0 \\ 0, & \exists f(i,j) = 0 \end{cases}$$

因此, 确定最小费用最大流的算法如下:

(1) 从零流开始, 令 $f_0 = 0$;

赋权: 当
$$f_k(i,j) < c(i,j)$$
,
$$\begin{cases} \omega^+(i,j) = a(i,j) \\ \Delta^+(i,j) = c(i,j) - f_k(i,j) \end{cases}$$
;

$$\stackrel{\cong}{=} f_k(i,j) = c(i,j), \qquad \begin{cases} \omega^+(i,j) = +\infty \\ \Delta^+(i,j) = 0 \end{cases};$$

$$\stackrel{\cong}{=} f_k(i,j) > 0, \qquad \begin{cases} \omega^-(i,j) = -a(i,j) \\ \Delta^-(i,j) = f_k(i,j) \end{cases};$$

$$\stackrel{\cong}{=} f_k(i,j) = 0, \qquad \begin{cases} \omega^-(i,j) = +\infty \\ \Delta^-(i,j) = 0 \end{cases}$$

(3)确定一条从 s 到 s' 的最短路 $R(s,s') = \{(s,i_1),(i_1,i_2),\cdots,(i_k,s')\}$.若 R(s,s') 的长度为+ ∞ ,则已得最小费用最大流、停机、否则转入(4).

(4)确定沿着该路 R(s,s') 的最大可增流量 $a=\min\{\Delta(s,i_1),\Delta(i_1,i_2),\cdots,\Delta(i_k,s')\}$,其中根据边的取向决定取 Δ^+ 或 Δ^- .

(5)生成新的流

$$f_{k+1}(i,j) = \begin{cases} f_k(i,j) + a, & (i,j)$$
为向前边
$$f_k(i,j) - a, & (i,j)$$
为向后边

若 $f_{k+1}(i,j)$ 已为最小费用最大流,停机,否则转入(2).

下面举例说明.

例 求如图 4-5 所示的最小费用最大流. 图中, 边旁的两个数字分别为 c(i,j)和 a(i,j).

从零开始, $f(i,j) \equiv 0$. 图 4-5 即可作为第一次迭代的赋权图(略去取 + ∞ 的权). 第一次迭代:

- (1)由最短路法,可得从s到s'的最短路为 $R(s,s') = \{(1,3),(3,5),(5,6)\}$,其费用为4.
- (2)确定沿R(s,s')的最大可增流量为

$$a = \min\{\Delta(1,3), \Delta(3,5), \Delta(5,6)\} = \min\{4,3,3\} = 3$$
.

(3)调整流量, 生成新的流

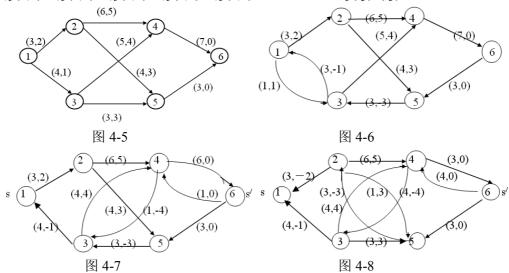
$$f(1,3) = 3$$
, $f(3,5) = 3$, $f(5,6) = 3$.

- (4)赋权, 得新的赋权图, 如图 4-6 所示. 其中, 边旁的两个数字分别为 $\triangle(i,j)$ 和 $\omega(i,j)$. 第二次迭代:
- (1)从s到s'的最短路为 $R(s,s') = \{(1,3),(3,4),(4,6)\}$,其中费用为 5.
- (2)沿 R(s,s') 的最大可增流量为 $a = \min\{\Delta(1,3),\Delta(3,4),\Delta(4,6)\} = \min\{1,5,7\} = 1$.
- (3)调整流量, 生成新的流 f(1,3)=4, f(3,4)=1, f(4,6)=1.
- (4)赋权、得新的赋权图、如图 4-7 所示.
- 第三次迭代:
- (1)从s到s'的最短路为 $R(s,s')=\{(1,2),(2,5),(5,3),(3,4),(4,6)\}$,费用为6.
- (2) R(s,s') 的最大可增流量

$$a = \min\{\Delta(1,2), \Delta(2,5), \Delta(5,3), \Delta(3,4), \Delta(4,6)\} = \min\{3,4,3,4,6\} = 3.$$

- (3)调整流量, 生成新的流 f(1,2)=3, f(2,5)=3, f(5,3)=0, f(3,4)=4, f(4,6)=4.
- (4)赋权得新的赋权图, 如图 4-8 所示.

至此, 在图 4-8 中, 不再存在从 s 到 s' 的最短路, 因此得最小费用的最大流为 f(1,2)=3, f(2,5)=3, f(3,4)=4, f(4,6)=4, f(1,3)=4, f(5,6)=3, f(2,4)=0, f(3,5)=0. 最小费用 $\sum f(i,j)a(i,j)=35$.



2. 运输问题

某产品有 m 个产地, 第 i 个产地的月最大产量和单位产品的成本分别为 c_i 单位/月和 $a_i(i=1,2,...,m)$. 该产品有 n 个需求地, 分别的需求量为 b_j 单位/月(j=1,2,...,n). 又已知从第 i 个产地到第 i 个需求地的最大可能运输量为 $\omega(i,j)$ 单位运价为 l(i,j). 问各地应生产多少产品? 如何调

运,才能尽可能满足需求,并使其总开支最省?

首先,我们虚设一个发点 s 和收点 s'. 从发点到第 i 个产地的权数设为 c_i ,单位费用为 $a_i(i=1,2,...,m)$. 从产地 i 到需求地 j 的权数 $\omega(i,j)$ 其单位费用为 l(i,j) (i=1,2,...,m, j=1,2,...,n),从需求地 j 到收点 s' 的权数为 b_i ,单位成本为零.这样就化为图 4-9 的网络的最小费用最大流问题.

运输问题化为最小费用最大流问题的技巧在于虚设一个发点和收点,以及确定各边相应的权数与单位费用.

另外, 上述运输问题也可表述为线性规划问题, 即下面的总费用最小的问题:

$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} l(i,j)g(i,j) \\ s.t. & \sum_{j=1}^{n} g(i,j) \le c_{i}, j = 1, 2, \dots, m \\ & \sum_{i=1}^{m} g(i,j) \ge b_{j}, j = 1, 2, \dots, n \\ & g(i,j) \ge 0, i = 1, 2, \dots, m, j = 1, 2, \dots, n \end{cases}$$

第一、二个约束条件蕴含着

$$\sum_{j=1}^{n} b_{j} \leq \sum_{i=1}^{m} \sum_{j=1}^{n} g(i,j) \leq \sum_{i=1}^{m} c_{i}.$$

即总需求量小于等于总供应量.

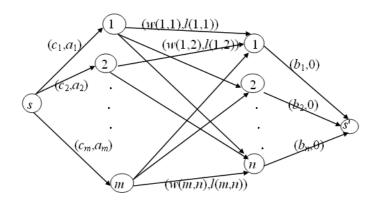


图 4-9

第五节 图的遍历

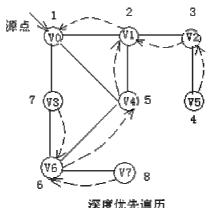
简单可这样说,从连通图的一顶点出发每边通过一次或每个顶点恰好通过一次.并不是每个连通图上都能实现的事;称图中含每条边的行迹为 Euler 行迹,闭的 Euler 行迹叫做 Euler 回路,含 Euler 回路的图叫做 Euler 图.

图的遍历有两种基本方法:

1、深度优先遍历

从图中指定的起点 v_0 出发(先访问 v_0)访问它的任意相邻接的顶点 v_1 , 再访问 v_1 的任一个未访问的相邻接顶点 v_2 , 如此下去, 直到某顶点已无被访问过的顶点, 则返回一步找前一个顶点的其他沿未被访问的邻接顶点, 重复以上过程直到所有的顶点都被访问. 例如, 按深度优先遍历法遍历下图, 遍

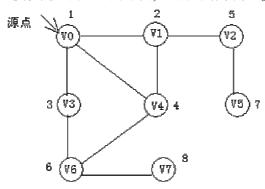
历的结果为: v₀ v₁ v₂ v₅ v₄ v₆ v₃ v₇.



深度优先遍历

2、广度优先遍历

从图指定的起点 v_0 出发,访问与它相邻的所有顶点 v_1, v_2, \ldots ,然后再依次访问 v_1, v_2, \ldots ,邻接的 尚未被访问的所有顶点, 再从这些顶点出发访问与它们相邻接的尚未被访问的顶点, 直到所有顶点 均被访问过为止. 例如, 按广度优先遍历法遍历下图, 遍历的结果为: v₀ v₁ v₃ v₄ v₂ v₆ v₅ v₇.



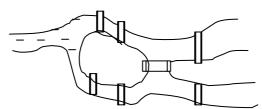
广度优先遍历

关于 Euler 遍历问题有下列的结论:

- (1)关于图是 Euler 图的充要条件.
- (2)连通图 G 有 Euler 行迹的充要条件是连通图 G 中至少有两个奇次点.

Euler 图解决了"七桥问题": 从河的一岸或一个岛上出法, 不重复地走遍七座桥.

这个问题的一个拓展问题就是"中国邮递员"问题: 邮递员每天要走过固定的区域内的所有街 道.



所谓欧拉回路与哥尼斯堡 7 桥问题相联系的. 在哥尼斯堡 7 桥问题中, 欧拉证明了不存在这样 的回路, 使它经过图中每条边且只经过一次又回到起始点. 与此相反, 设 G(V, E)为一个图, 若存在一 条回路, 使它经过图中每条边且只经过一次又回到起始点, 就称这种回路为欧拉回路, 并称图 G 为 欧拉图.

在一个图中, 连接一个节点的边数称为该节点的度数. 对欧拉图, 我们有下列结果:

定理 1 对连通图 G(V, E), 下列条件是相互等价的:

- (1)G是一个欧拉图;
- (2)G的每一个节点的度数都是偶数;
- (3)G 的边集合 E 可以分解为若干个回路的并.

证明: $(1) \rightarrow (2)$ 已知 G 为欧拉图,则必存在一个欧拉回路.回路中的节点都是偶度数.

- (2) \rightarrow (3) 设 G 中每一个节点的度数均为偶数. 若能找到一个回路 C_1 使 $G=C_1$,则结论成立. 否则,令 $G_1=G\setminus C_1$,由 C_1 上每个节点的度数均为偶数,则 G_1 中的每个节点的度数亦均为偶数. 于是在 G_1 必存在另一个回路 C_2 . 令 $G_2=G_1\setminus C_2$,……,由于 G 为有限图,上述过程经过有限步,最后必得一个回路 C_r 使 $G_r=G_{r-1}\setminus C_r$ 上各节点的度数均为零,即 $C_r=G_{r-1}$. 这样就得到 G 的一个分解 $G=C_1\cup C_2\cup\ldots\cup C_r$.
- (3)→(1)设 $G=C_1 \cup C_2 \cup ... \cup C_r$,其中 $C_i=(i=1,2,\cdots,r)$ 均为回路。由于 G 为连通图,对任意回路 C_i 必存在另一个回路 C_j 与之相连,即 C_i 与 C_j 存在共同的节点. 现在我们从图 G 的任意节点出发,沿着所在的回路走,每走到一个共同的节点处,就转向另一个回路,...,这样一直走下去就可走遍 G 的每条边且只走过一次,最后回到原出发节点,即 G 为一个欧拉图.

利用定理 1 去判断一个连通图是否为欧拉图比较容易, 但要找出欧拉回路, 当连通图比较复杂时就不太容易了. 下面介绍一种求欧拉回路的算法.

弗罗莱算法

算法步骤如下:

- (1)任取起始点 $v_0, v_0 \rightarrow R$.
- (2)设路 $R = \{e_1(v_0, v_{i_1}), \{e_2(v_{i_1}, v_{i_2}), \dots, \{e_r(v_{i_{r-1}}, v_{i_r})\}$ 已选出,则从 $E \setminus \{e_1, e_2, \dots e_r, \}$ 中选出边 e_{r+1} ,使 e_{r+1} 与 v_i 相连,除非没有其它选择, $G_r \setminus \{e_{r+1}\}$ 仍应为连通的.
 - (3)重复步骤(2), 直至不能进行下去为止.
- **定理 2** 连通的有向图存在欧拉回路的充分必要条件是对任意节点,进入该节点边数(进数)与 离开该点的边数(出数)相等.

第五章 数学软件使用初步

在历届的数学建模竞赛中,参赛者对数学软件的熟练掌握程度对建模的成功与失败起着非常重要的作用.从数据的处理到公式的推导,从模型的建立到模型的求解,没有数学软件的帮助几乎是不可能完成的.从今后的趋势来看,数学软件在建模过程中会占据更为重要的地位.现在数学软件能做的事情已经非常之多,几乎涉及到了所有的数学领域,数的运算、公式推导、方程求解、微积分、线性代数、概率统计、规划求解、函数作图、图论、数论、数值计算、信号处理等等.随着计算技术的不断发展,数学软件的功能会更强大、更智能.

当今, 综合性的数学软件主要有 MATLAB, MAPLE, Mathematica, MathCAD 等, 专用数学软件主要有 SPSS, LINDO, LINGO, 这些在建模竞赛过程中都可能用到. 在这里, 我们主要介绍 MATLAB、MAPLE、LINDO 和 LINGO.

第一节 MATLAB

§ 1.1 MATLAB 简介

MATLAB 名字是由 MATrix 和 LABoratory 两个词的前三个字母组合而成的. 它是 MathWorks 公司于 1982 年推出的一套高性能的数值计算和可视化数学软件. 由于使用编程运算与人进行科学计算的思路和表达方式完全一致, 所以不象学习其它高级语言如 Basic、fortran 和 C 等那样难于掌握. 用 Matlab 编写程序犹如在演算纸上排列出公式与求解问题, 所以又被称为演算纸式科学算法语言. 在这个环境下, 对所要求解的问题, 用户只需简单地列出数学表达式, 其结果便以数值或图形方式显示出来.

MATLAB 的含义是矩阵实验室,主要用于方便矩阵的存取,其基本元素是无须定义维数的矩阵.自问世以来,就以数值计算称雄. MATLAB 进行数值计算的基本单位是复数数组(或称阵列),这使得MATLAB 高度"向量化". 经过十几年的完善和扩充,现已发展成为线性代数课程的标准工具. 由于它不需定义数组的维数,并给出矩阵函数、特殊矩阵专门的库函数,使之在求解诸如信号处理、建模、系统识别、控制、优化等领域的问题时,显得大为简捷、高效、方便,这是其它高级语言所不能比拟的. 美国许多大学的实验室都安装有供学习和研究之用. 在那里, MATLAB 是攻读学位的大学生、硕士生、博士生必须掌握的基本工具.

MATLAB 中包括了被称作工具箱(TOOLBOX)的各类应用问题的求解工具. 工具箱实际上是对 MATLAB 进行扩展应用的一系列 MATLAB 函数(称为 M 文件),它可用来求解各类学科的问题,包括信号处理、图象处理、控制系统辨识、神经网络等. 随着 MATLAB 版本的不断升级,其所含的工具箱的功能也越来越丰富,因此,应用范围也越来越广泛,成为涉及数值分析的各类工程师不可不用的工具.

§ 1.2 基本运算与函数

在MATLAB下进行基本数学运算,只需将运算式直接打入提示号(>>)之后,并按入Enter键即可. 例如:

>>(5*2+1.3-0.8)*10/25

ans =

4.2000

MATLAB 会将运算结果直接存入一变量 ans, 代表 MATLAB 运算后的答案(Answer), 并显示其数值于屏幕上. (为简便起见, 在下述各例中, 我们不再写出 MATLAB 的提示号.)

★小提示

">>"是 MATLAB 的提示符号(Prompt), 但在 PC 中文视窗系统下, 由于编码方式不同, 此提示符号常会消失不见, 但这并不会影响到 MATLAB 的运算结果.

我们也可将上述运算式的结果设定给另一个变量 x:

 $x = (5*2+1.3-0.8)*10^2/25$

 $_{\mathrm{X}} =$

42

此时 MATLAB 会直接显示 x 的值. 由上例可知, MATLAB 认识所有一般常用到的加(+)、减(-)、乘(*)、除(/)的数学运算符号, 以及幂次运算(^).

★小提示

MATLAB 在使用变量之前可以不事先经过变量声明(Variable declaration).

若不想让 MATLAB 每次都显示运算结果,只需在运算式最后加上分号";"即可,如下例: $y = \sin(10)*\exp(-0.3*4^2)$;

若要显示变量 y 的值, 直接键入 y 即可:

y y =

-0.0045

在上例中, sin 是正弦函数, exp 是指数函数, 这些都是 MATLAB 常用到的数学函数. 下表即为 MATLAB 常用的基本数学函数及三角函数:

小整理 MATLAB 常用的基本数学函数

	三年 111111111111111111111111111111111111
abs(x)	实数的绝对值或复数的模
angle(z)	复数 z 的相角(Phase angle)
sqrt(x)	开平方
real(z)	复数 z 的实部
imag(z)	复数 z 的虚部
conj(z)	复数 z 的共轭复数
round(x)	四舍五入至最近整数
fix(x)	无论正负, 舍去小数至最近整数
floor(x)	大于 x 的最近整数
ceil(x)	小于 x 的最近整数
rat(x)	将实数 x 化为分数表示
rats(x)	将实数 x 化为多项分数展开
sign(x)	符号函数 (Signum function)
	当 x<0 时, sign(x)=-1;
	当 x=0 时, sign(x)=0;
	当 x>0 时, sign(x)=1.
rem(x,y)	求x除以y的余数
gcd(x,y)	整数 x 和 y 的最大公因数
lcm(x,y)	整数 x 和 y 的最小公倍数
exp(x)	自然指数 e ^x

pow2(x)	2 的指数 2 ^x
log(x)	以 e 为底的对数, 即自然对数或 ln(x)
log2(x)	以 2 为底的对数 log ₂ (x)
log10(x)	以 10 为底的对数 log ₁₀ (x)

小整理 MATLAB 常用的三角函数

sin(x)	正弦函数
cos(x)	余弦函数
tan(x)	正切函数
asin(x)	反正弦函数
acos(x)	反余弦函数
atan(x)	反正切函数
atan2(x,y)	四象限的反正切函数
sinh(x)	双曲正弦函数
cosh(x)	双曲余弦函数
tanh(x)	双曲正切函数
asinh(x)	反双曲正弦函数
acosh(x)	反双曲余弦函数
atanh(x)	反双曲正切函数

变量也可用来存放向量或矩阵,并进行各种运算,如下例的行向量(Row vector)运算:

 $x = [1 \ 3 \ 5 \ 2];$

y = 2*x+1

y =

3 7 11 5

★小提示

变量命名的规则:第一个字母必须是英文字母,字母间不可留空格,最多只能有 19 个字母,MATLAB 会忽略多余字母.

我们可以随意更改、增加或删除向量的元素:

y(3)=2 %更改第三个元素

y =

3725

y(6) = 10 %加入第六个元素

y =

3725010

y(4) = [] %删除第四个元素,

y =

372010

在上例中, MATLAB 会忽略所有在百分比符号(%)之后的文字, 视为程序的注解(Comments). MATLAB 亦可取出向量的一个元素或一部份来做运算:

x(2)*3+y(4) %取出 x 的第二个元素和 y 的第四个元素来做运算

ans =

g

y(2:4)-1 %取出 y 的第二至第四个元素来做运算

ans =

61-1

```
在上例中, 2:4 代表一个由 2、3、4 组成的向量,同样的方法可用于产生公差为 1 的等差数列:
x = 7:16
x =
```

7 8 9 10 11 12 13 14 15 16

若不希望公差为 1,则可将所需公差直接置于 4 与 13 之间:

x = 7:3:16 %公差为 3 的等差数列

 $\mathbf{x} =$

7 10 13 16

事实上, 我们可利用 linspace 来产生任意的等差数列:

x = linspace(4, 10, 6) %等差数列: 首项为 4,末项为 10,项数为 6

 $_{\rm X} =$

4.0000 5.2000 6.4000 5.6000 8.8000 10.0000

若对 MATLAB 函数用法有疑问, 可随时使用 help 来寻求在线帮助(on-line help).

将列向量转置(Transpose)后,即可得到行向量(Column vector):

z = x'

z =

4.0000

5.2000

6.4000

5.6000

8.8000

10.0000

不论是行向量或列向量, 我们均可用相同的函数找出其元素个数、最大值、最小值等: length(z) %z 的元素个数

ans =

max(z) %z 的最大值

ans =

10

min(z) %z 的最小值

ans =

4

小整理 话用于向量的常用函数有

	7. 定性 旭川1 四里的印川四数月
min(x)	向量 x 的元素的最小值
max(x)	向量 x 的元素的最大值
mean(x)	向量 x 的元素的平均值
median(x)	向量 x 的元素的中位数
std(x)	向量 x 的元素的标准差
diff(x)	向量 x 的相邻元素的差
sort(x)	对向量 x 的元素进行排序(Sorting)
length(x)	向量 x 的元素个数
norm(x)	向量 x 的欧氏(Euclidean)长度
sum(x)	向量 x 的元素总和
prod(x)	向量 x 的元素总乘积

cumsum(x)	向量 x 的累计元素总和
cumprod(x)	向量 x 的累计元素总乘积
dot(x, y)	向量 x 和 y 的内积
cross(x, y)	向量x和y的外积

(大部份的向量函数也可适用于矩阵)

若要输入矩阵,则必须在每一列结尾加上分号(;),如下例:

 $A = [1 \ 2 \ 3 \ 4; 5 \ 6 \ 7 \ 8; 9 \ 10 \ 11 \ 12];$

Α

A =

同样地, 我们可以对矩阵进行各种处理:

A(2,3)=5 %改变位于第二列,第三行的元素值

A =

B=A(2,1:3) %取出部份矩阵 B

B =

A=[AB'] %将B转置后以行向量并入A

A =

A(:, 2) = [] %删除第二行(: 代表所有列)

A =

A=[A;4321] %加入第四列

A =

A([14],:)=[] %删除第一和第四列(:代表所有行)

A =

这几种矩阵处理的方式可以相互迭代运用,产生各种意想不到的效果,就看各位的巧思和创意.

★小提示

在 MATLAB 的内部资料结构中,每一个矩阵都是一个以行为主(Column-oriented)的阵列(Array)

因此对于矩阵元素的存取,我们可用一维或二维的索引(Index)来定址.举例来说,在上述矩阵 A 中,位于第二列、第三行的元素可写为 A(2,3) (二维索引)或 A(6)(一维索引,即将所有直行进行堆叠后的第六个元素).

此外, 若要重新安排矩阵的形状, 可用 reshape 命令:

B = reshape(A, 4, 2) %4 是新矩阵的列数, 2 是新矩阵的行数

B =

5 8

9 12

5 6

11 5

★小提示

A(:)就是将矩阵 A 每一列堆叠起来, 成为一个行向量, 而这也是 MATLAB 变量的内部储存方式. 以前例而言, reshape(A, 8, 1)和 A(:)同样都会产生一个 8×1 的矩阵..

MATLAB 可在同时执行数个命令, 只要以逗号或分号将命令隔开:

 $x = \sin(pi/3); y = x^2; z = y^{10},$

z =

5.5000

若一个数学运算是太长,可用三个句点将其延伸到下一行:

 $z = 10*\sin(pi/3)*...$

 $\sin(pi/3)$;

MATLAB 有些永久常数(Permanent constants), 使用者可直接取用, 例如:

pi

ans =

3.1416

下表即为 MATLAB 常用到的永久常数.

小整理 MATLAB 的永久常数

i或j	基本虚数单位(即 $\sqrt{-1}$)
eps	系统的浮点(Floating-point)精确度
inf	无限大,例如 1/0
nan 或 NaN	非数值(Not a number), 例如 0/0
pi	圆周率(= 3.1415926)
realmax	系统所能表示的最大数值
realmin	系统所能表示的最小数值
nargin	函数的输入参数个数
nargin	函数的输出参数个数

§ 1.3 基本平面绘图命令

MATLAB 不但擅长于矩阵相关的数值运算,也适合用在各种科学可视化表示(Scientific visualization). 本节将介绍 MATLAB 基本 xy 平面及 xyz 空间的各项绘图命令,包含一维曲线及二维曲面的绘制、输出及存档.

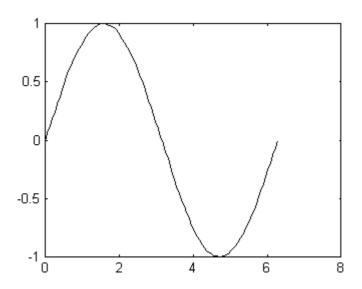
plot 是绘制一维曲线的基本函数, 但在使用此函数之前, 我们需先定义曲线上每一点的 x 及 y 座

标. 下例可画出一条正弦曲线:

close all; x=linspace(0, 2*pi, 100); %100 个点的 x 座标

y=sin(x); %对应的 y 座标

plot(x,y);



小整理: MATLAB 基本绘图函数

plot	x 轴和 y 轴均为线性刻度(Linear scale)
loglog	x 轴和 y 轴均为对数刻度(Logarithmic scale)
semilogx	x 轴为对数刻度, y 轴为线性刻度
semilogy	x 轴为线性刻度, y 轴为对数刻度

若要画出多条曲线,只需将坐标对依次放入 plot 函数即可:

plot(x, sin(x), x, cos(x));

若要改变颜色, 在座标对后面加上相关字串即可:

plot(x, sin(x), 'c', x, cos(x), 'g');

若要同时改变颜色及图线型态(Line style), 也是在座标对后面加上相关字串即可: plot(x, sin(x), 'co', x, cos(x), 'g*');

小整理 plot 绘图函数的参数

字元	颜色	字元	图线型态
y	黄色		点
k	黑色	0	圆
W	白色	X	X
b	蓝色	+	+
g	绿色	*	*
r	红色	-	实线
c	亮青色	:	点线
m	锰紫色		点虚线
			虚线

图形完成后, 我们可用 axis([xmin,xmax,ymin,ymax])函数来调整图轴的范围: axis([0, 6, -1.2, 1.2]);

此外, MATLAB 也可对图形加上各种注解与处理:

xlabel('Input Value'); %x 轴注解

```
ylabel('Function Value'); %y 轴注解
title('Two Trigonometric Functions'); %图形标题
legend('y = sin(x)','y = cos(x)'); %图形注解
text(x,y, 'str'); %在坐标(x,y)处显示字符串 str
grid on; %显示格线
我们可用 subplot 来同时画出数个小图形于同一个视窗之中:
subplot(2,2,1); plot(x, sin(x));
subplot(2,2,2); plot(x, cos(x));
subplot(2,2,3); plot(x, sinh(x));
subplot(2,2,4); plot(x, cosh(x));
MATLAB 还有其他各种二维绘图函数,以适合不同的应用,详见下表.
```

小整理: 其他各种二维绘图函数

7. 是在,		
bar	长条图	
errorbar	图形加上误差范围	
fplot	较精确的函数图形	
polar	极座标图	
hist	累计图	
rose	极座标累计图	
stairs	阶梯图	
stem	针状图	
fill	实心图	
feather	羽毛图	
compass	罗盘图	
quiver	向量场图	

§ 1.4 基本三维绘图命令

```
plot3 命令将绘制二维图形的函数 plot 的特性扩展到三维空间. 函数格式除了包括第三维的信息 (比如 Z 方向)之外,与二维函数 plot 相同. plot3 一般语法调用格式是 plot3(x1,y1,z1,S1,x2,y2,z2,S2,…)
```

这里 xn,yn 和 zn 是向量或矩阵, Sn 是可选的字符串, 用来指定颜色、标记符号和线形.

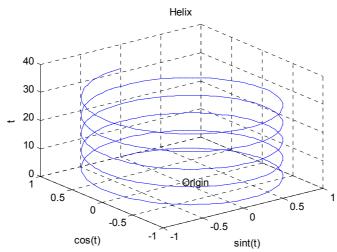
总的来说, plot3 可用来画一个单变量的三维函数. 如下为一个三维螺旋线例子:

40

```
总的来说, plot3 可用来画一个甲变量的三维函数.
t=0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
title('Helix'),xlabel('sint(t)'),ylabel('cos(t)'),zlabel('t')
text(0,0,0, 'Origin')
grid
v = axis
v =
```

1 -1 1 0

-1



从上例可明显看出,二维图形的所有基本特性在三维中仍都存在. axis 命令扩展到三维,只是返回 Z 轴界限(0 和 40),在数轴向量中增加两个元素. 函数 zlabel 用来指定 z 轴的数据名称,函数 grid 在图底绘制三维网格. 函数 test(x,y,z, 'string')在由三维坐标 x,y,z 所指定的位置放一个字符串. 另外,子图和多图形窗口可以直接应用到三维图形中.

通过指定 plot3 命令的多个参数或使用 hold 命令,可以把多条直线或曲线重叠画出. 例如,增加维数的 plot3 命令可以使多个二维图形沿一个轴排列起来,而不是直接将二维图形叠到另一个的上面.

```
x=linspace(0,3*pi); % x-axis data

z1=sin(x); % plot in x-z plane

z2=sin(2*x);

z3=sin(3*x);

y1=zeros(size(x)); % spread out along y-axis

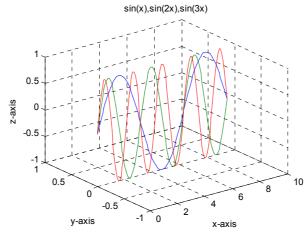
y3=zeros(size(x)); % by giving each diffent y-axis values

y2=y3/2;

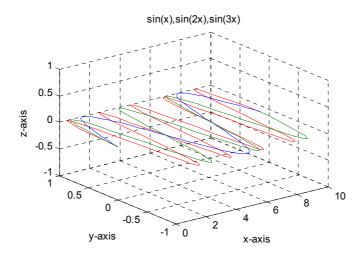
plot3(x,y1,z1,x,y2,z2,x,y3,z3);

grid,xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis')

title('sin(x),sin(2x),sin(3x)')
```



上述图形也可以沿另外方向堆列. plot3(x,z1,y1,x,z2,y2,x,z3,y3) grid,xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis') title('sin(x),sin(2x),sin(3x)')



§ 1.5 流程控制

计算机编程语言和可编程计算器提供许多功能,它允许你根据决策结构控制命令执行流程.如果你以前已经使用过这些功能,对此就会很熟悉.相反,如果不熟悉控制流,本章材料初看起来或许复杂些.如果这样,就慢慢来.

控制流极其重要,因为它使过去的计算影响将来的运算. MATLAB 提供三种决策或控制流结构. 它们是: for 循环, while 循环和 if-else-end 结构. 由于这些结构经常包含大量的 MATLAB 命令, 故经常出现在 M 文件中,而不是直接加在 MATLAB 提示符下.

5.1 for 循环

for 循环允许一组命令以固定的和预定的次数重复. for 循环的一般形式是:

for x = array {commands}

end

在 for 和 end 语句之间的 $\{commands\}$ 按数组中的每一列执行一次. 在每一次迭代中, x 被指定为数组的下一列, 即在第 n 次循环中, x=array $\{c, n\}$. 例如,

for n=1:10

 $x(n)=\sin(n*pi/10);$

end

X

x =

Columns 1 through 7

Columns 8 through 10

换句话,第一语句是说:对 n 等于 1 到 10,求所有语句的值,直至下一个 end 语句.第一次通过 for 循环 n=1,第二次 n=2,如此继续,直至 n=10.在 n=10以后,for 循环结束,然后求 end 语句后面的任何命令值,在这种情况下显示所计算的 x 的元素.

for 循环的其它重要方面是:

1. for 循环不能用 for 循环内重新赋值循环变量 n 来终止.

for n=1:10

```
x(n)=\sin(n*pi/10);
        n=10;
     end
    X
    \mathbf{x} =
      Columns 1 through 7
        0.3090
                  0.5878
                            0.8090
                                      0.9511
                                             1.0000
                                                        0.9511
                                                                    0.8090
      Columns 8 through 10
        0.5878
                  0.3090
                            0.0000
    2. 语句 1:10 是一个标准的 MATLAB 数组创建语句. 在 for 循环内接受任何有效的 MATLAB 数
组.
    data=[3 9 45 6; 7 16 -1 5]
    data =
         3
               9
                     45
                            6
         7
              16
                     -1
                            5
    for n=data
         x=n(1)-n(2)
    end
    _{\rm X} =
        -4
    _{\rm X} =
        -7
    _{\rm X} =
        46
    _{\rm X} =
         1
    3. for 循环可按需要嵌套.
    for n=1:5
       for m=5:-1:1
          A(n,m)=n^2+m^2;
       end
       disp(n)
    end
         1
         2
         3
         4
         5
    Α
    A =
         2
               5
                     10
                           17
                                 26
         5
               8
                           20
                                 29
                     13
        10
              13
                     18
                           25
                                 34
```

17

20

25

32

41

```
26 29 34 41 50
```

4. 当有一个等效的数组方法来解给定的问题时, 应避免用 for 循环. 例如, 上面的第一个例子可被重写为

```
n=1:10;

x=sin(n*pi/10)

x = 

Columns 1 through 7

0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090

Columns 8 through 10

0.5878 0.3090 0.0000
```

两种方法得出同样的结果, 而后者执行更快, 更直观, 要求较少的输入.

5. 为了得到最大的速度, 在 for 循环(while 循环)被执行之前, 应预先分配数组. 例如, 前面所考虑的第一种情况, 在 for 循环内每执行一次命令, 变量 x 的大小增加 1. 迫使 MATLAB 每通过一次循环要花费时间对 x 分配更多的内存. 为了消去这个步骤, for 循环的例子应重写为

```
x=zeros(1,10); % preallocated memory for x for n=1:10  x(n)=\sin(n*pi/10);  end
```

现在, 只有 x(n)的值需要改变.

5.2 while 循环

与 for 循环以固定次数求一组命令的值相反, while 循环以不定的次数求一组语句的值. while 循环的一般形式是:

```
while expression {commands} end
```

只要在表达式里的所有元素为真,就执行 while 和 end 语句之间的 {commands}. 通常,表达式的求值给出一个标量值,但数组值也同样有效. 在数组情况下,所得到数组的所有元素必须都为真. 考虑下列例子:

```
num=0;EPS=1;
while (1+EPS)>1
EPS=EPS/2;
num=num+1;
end
num
num = 53
EPS=2*EPS
EPS = 2.2204e-016
```

这个例子表明了计算特殊 MATLAB 值 EPS 的一种方法,它是一个可加到 1,而使结果以有限精度大于 1 的最小数值.这里我们用大写 EPS,因此 MATLAB 的 EPS 的值不会被覆盖掉.在这个例子里, EPS 以 1 开始.只要(1+EPS)>1 为真(非零),就一直求 while 循环内的命令值.由于 EPS 不断地被 2 除, EPS 逐渐变小以致于 EPS+1 不大于 1. (记住,发生这种情况是因为计算机使用固定数的数值来表示数. MATLAB 用 16 位,因此,我们只能期望 EPS 接近 10-16.) 在这一点上,(1+EPS)>1 是假(零),

于是 while 循环结束. 最后, EPS 与 2 相乘, 因为最后除 2 使 EPS 太小.

5.3 if-else-end 结构

很多情况下,命令的序列必须根据关系的检验有条件地执行. 在编程语言里,这种逻辑由某种 if-else-end 结构来提供. 最简单的 if-else-end 结构是:

```
if expression {commands} end
```

如果在表达式中的所有元素为真(非零),那么就执行 if 和 end 语言之间的 {commands}.在表达式包含有几个逻辑子表达式时,即使前一个子表达式决定了表达式的最后逻辑状态,仍要计算所有的子表达式.例如,

```
apples=10;
                    % number of apples
                    % cost of apples
cost=apples*25
cost =
   250
if apples>5
                    % give 20% discount for larger purchases
    cost = (1-20/100)*cost;
end
cost
cost =
   200
假如有两个选择, if-else-end 结构是:
if expression
    commands evaluated if True
else
    commands evaluated if False
end
在这里, 如果表达式为真, 则执行第一组命令; 如果表达式是假, 则执行第二组命令.
当有三个或更多的选择时, if-else-end 结构采用形式
if expression1
   commands evaluated if expression1 is True
elseif expression2
   commands evaluated if expression2 is True
elseif expression3
   commands evaluated if expression3 is True
elseif expression4
   commands evaluated if expression4 is True
elseif ······
    . . . . . .
else
   commands evaluated if no other expression is True
end
```

最后的这种形式,只和所碰到的、与第一个真值表达式相关的命令被执行;接下来的关系表达式不检验,跳过其余的 if-else-end 结构. 而且,最后的 else 命令可有可无.

现在我们知道了如何用 if-else-end 结构来决策, 就有可能提出一种合理的方法来跳出或中断 for

```
循环和 while 循环.
EPS=1;
for num=1:1000
EPS=EPS/2;
if (1+EPS)<=1
EPS=EPS*2
break
end
end
EPS =
2.2204e-016
num
num =
53
```

这个例子演示了估算 EPS 的另一种方法. 在这种情况下, for 循环构造成要执行足够多的次数. if-else-end 结构检验要看 EPS 是否变得足够小. 如果是, EPS 乘 2, break 命令强迫 for 循环提早结束, num=53.

在这个例子里, 当执行 break 语句时, MATLAB 跳到循环外下一个语句. 在现在情况下, 它返回到 MATLAB 的提示符并显示 EPS. 如果一个 break 语句出现在一个嵌套的 for 循环或 while 循环结构里, 那么 MATLAB 只跳出 break 所在的那个循环, 不跳出整个嵌套结构.

5.4 小结

MATLAB 控制流功能可以概括如下:

控制流	结构
for x = array	for 循环, 每一次迭代将 x 赋以数组的
commands	第 i 列, 执行命令
end	
while expression	while 循环, 只要表达式的所有元素
commands	为真或非零, 执行命令.
end	
if expression	简单的 if-else-end 结构, 若在表达式
commands	中的所有元素是真值或非零, 执行
end	命令.
if expression	具有两条路径的 if-else-end 结构,
commands evaluated if True	若表达式为真或非零, 则执行一组
else expression	命令. 若表达式为假或零,
commands evaluated if False	则执行另一组命令.
end	
if expression1	
commands evaluated if expression1 is True	
elseif expression2	
commands evaluated if expression2 is True	最一般的 if-else-end 结构.
if expression3	只执行与第一个真值表达式相关
commands evaluated if expression3 is True	的命令.
elseif ······	

else
commands evaluated if no other
expression is True
end

break

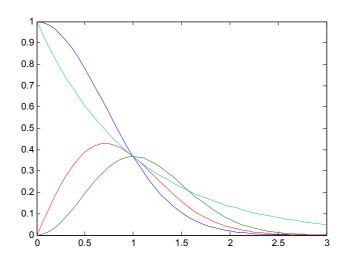
结束 for 循环和 while 循环的执行

§ 1.6 M 文件

Matlab 允许将一系列将要运行的语句事先存放在后缀名为.m 的文件中, 然后在 Matlab 命令行中输入该文件的主文件名即可. 当然, 要让 Matlab 能过文件名找得到文件并执行相应的语句, 必须把文件放到 Matlab 的 Current Directory 中, 该目录可以在 Matlab 界面中自己设定, 默认是安装目录下的 work 目录. M 文件是普通的文本文件, 可以用记事本或其它的文本编辑工具编写. Matlab 提供了 M 文件编辑工具, 可通过菜单 File\new\M-file 新建一个 M 文件, 也可以用 File\Open 打开一个已有的 M 文件进行修改.

编写 M 文件有两个目的. 一是将多个 Matlab 语句集成在一个文件, 然后通过一个命令一次运行. 这相当于批处理文件. 例如用记事本输入以下内容, 并以 test.m 为文件名保存到 Matlab 的 Current Directory 中:

```
n=50;
b=3;
a=0;
x=linspace(a,b,n);
e1=exp(-x.^2);
e2=(x.^2).*exp(-x.^2);
e3=x.*exp(-x.^2);
e4=exp(-x);
plot(x,e1,x,e2,x,e3,x,e4)
然后在 Matlab 命令行中 test, 可得到如下图形:
```



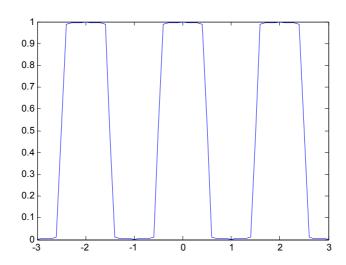
二是编写自定义函数. Matlab 的自定义函数以 M 文件形式存储在计算机中,一个函数用一个 M 文件存储, 这类 M 文件称为函数 M 文件. 函数 M 文件的一般格式:

function 函数值=函数名(参数)

函数体

function 是关键字,表明下面定义一个函数. 函数值用来取得函数的返回值,通常是在函数体中对其进行赋值,一个函数可以有多个函数值,多个函数值用逗号隔开,有时候也称函数值为输出参数. 函数名必须与 M 文件的主文件名相同,是 Matlab 调用该函数的依据,函数名最多可以有 31 个字符,并且与操作系统有关,有些操作系统允许的字符数可能更少,其命名规则与变量命名规则相同.参数用来从函数外部获取数据,可以有 0 个或多个参数,多个参数之间用逗号隔开,有时候也称参数为输入参数. 函数体是实现该函数的功能的语句,当最后一行被执行完毕或遇到 return 语句时退出函数,并返回到 Matlab 命令行. 例如,在记事本中输入以下内容,并以文件名 sqpulse.m 存到 Matlab 的 Current Directory 指定的目录中:

```
function f=sqpulse(n,x)
% 递归函数用于求和
% 1/2+2/pi cos(x pi) + ... +2sin(n pi/2) cos(n x pi)
% 当 n-->inf 这将等于阶跃的平方
if(n==1)
    f=1/2+2/pi*cos(x*pi);
else
    f=2*sin(n*pi/2)/n/pi*cos(n*x*pi)+sqpulse(n-1,x);
end
然后在 Matlab 命令行中输入
n=100;
x=-3:0.1:3;
plot(x,sqpulse(n,x)) %画出函数 sqpulse 的图形
可得到如下图形:
```

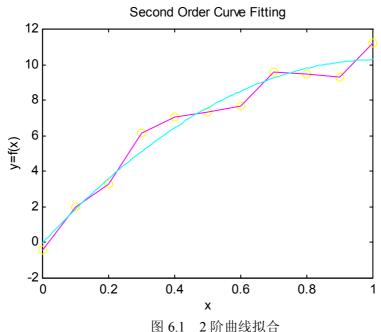


§ 1.7 曲线拟合与插值

在大量的应用领域中,人们经常面临用一个解析函数描述数据(通常是测量值)的任务. 对这个问题有两种方法. 在插值法里,数据假定是正确的,要求以某种方法描述数据点之间所发生的情况. 这种方法在下一节讨论. 这里讨论的方法是曲线拟合或回归. 人们设法找出某条光滑曲线,它最佳地拟合数据,但不必要经过任何数据点. 图 6.1 说明了这两种方法. 标有'o'的是数据点;连接数据点的实线描绘了线性内插,虚线是数据的最佳拟合.

6.1 曲线拟合

曲线拟合涉及回答两个基本问题:最佳拟合意味着什么?应该用什么样的曲线?可用许多不同的方法定义最佳拟合,并存在无穷数目的曲线.所以,从这里开始,我们走向何方?正如它证实的那样,当最佳拟合被解释为在数据点的最小误差平方和,且所用的曲线限定为多项式时,那么曲线拟合是相当简捷的.数学上,称为多项式的最小二乘曲线拟合.如果这种描述使你混淆,再研究图 6.1.虚线和标志的数据点之间的垂直距离是在该点的误差.对各数据点距离求平方,并把平方距离全加起来,就是误差平方和.这条虚线是使误差平方和尽可能小的曲线,即是最佳拟合.最小二乘这个术语仅仅是使误差平方和最小的省略说法.



在 MATLAB 中, 函数 polyfit 求解最小二乘曲线拟合问题. 为了阐述这个函数的用法, 让我们以上面图 6.1 中的数据开始.

x=[0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];

为了用 polyfit, 我们必须给函数赋予上面的数据和我们希望最佳拟合数据的多项式的阶次或度. 如果我们选择 n=1 作为阶次, 得到最简单的线性近似. 通常称为线性回归. 相反, 如果我们选择 n=2 作为阶次, 得到一个 2 阶多项式. 现在, 我们选择一个 2 阶多项式.

n=2; % polynomial order

p=polyfit(x, y, n)

p =

-9.8108 20.1293 -0.0317

polyfit 的输出是一个多项式系数的行向量. 其解是 y = -9.8108x2 + 20.1293x - 0.0317. 为了将曲线拟合解与数据点比较、让我们把二者都绘成图.

xi=linspace(0, 1, 100); % x-axis data for plotting

z=polyval(p, xi);

为了计算在 xi 数据点的多项式值,调用 MATLAB 的函数 polyval.

plot(x, y, 'o', x, y, xi, z, ':')

画出了原始数据 x 和 y, 用'o'标出该数据点, 在数据点之间, 再用直线重画原始数据, 并用点':'线, 画出多项式数据 xi 和 z.

xlabel('x'), ylabel('y=f(x)'), title('Second Order Curve Fitting')

将图作标志. 这些步骤的结果表示于前面的图 6.1 中.

多项式阶次的选择是有点任意的. 两点决定一直线或一阶多项式. 三点决定一个平方或 2 阶多项式. 按此进行, n+1 数据点唯一地确定 n 阶多项式. 于是, 在上面的情况下, 有 11 个数据点, 我们可选一个高达 10 阶的多项式. 然而, 高阶多项式给出很差的数值特性, 人们不应选择比所需的阶次高的多项式. 此外, 随着多项式阶次的提高, 近似变得不够光滑, 因为较高阶次多项式在变零前, 可多次求导. 例如, 选一个 10 阶多项式

pp=polyfit(x, y, 10);

format short e % change display format

pp.' % display polynomial coefficients as a column

ans =

- -4.6436e+005
- 2.2965e+006
- -4.8773e+006
- 5.8233e+006
- -4.2948e+006
- 2.0211e+006
- -6.0322e+005
- 1.0896e+005
- -1.0626e+004
- 4.3599e+002
- -4.4700e-001

要注意在现在情况下,多项式系数的规模与前面的 2 阶拟合的比较.还要注意在最小 (-4.4700e-001)和最大(5.8233e+006)系数之间有 7 个数量级的幅度差.将这个解作图,并把此图与原始数据及 2 阶曲线拟合相比较,结果如何呢?

zz=polyval(pp, xi);

% evaluate 10th order polynomial

plot(x, y, 'o', xi, z, ':', xi, zz)

% plot data

xlabel('x'), ylabel('y=f(x)'), title('2nd and 10th Order curve Fitting')

在下面的图 6.2 中,原始数据标以'o',2 阶曲线拟合是虚线,10 阶拟合是实线.注意,在10 阶拟合中,在左边和右边的极值处,数据点之间出现大的纹波. 当企图进行高阶曲线拟合时,这种纹波现象经常发生. 根据图 6.2,显然,"越多就越好"的观念在这里不适用.

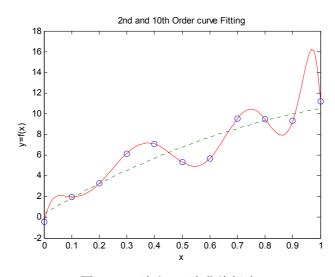


图 6.2 2 阶和 10 阶曲线拟合

6.2 一维插值

正如在前一节对曲线拟合所描述的那样,插值定义为对数据点之间函数的估值方法,这些数据点是由某些集合给定.当人们不能很快地求出所需中间点的函数值时,插值是一个有价值的工具.例如,当数据点是某些实验测量的结果或是过长的计算过程时,就有这种情况.

或许最简单插值的例子是MATLAB的作图. 按缺省, MATLAB用直线连接所用的数据点以作图. 这个线性插值猜测中间值落在数据点之间的直线上. 当然, 当数据点个数的增加和它们之间距离的减小时, 线性插值就更精确. 例如,

x1=linspace(0, 2*pi, 60); x2=linspace(0, 2*pi, 6); plot(x1, sin(x1), x2, sin(x2), '-') xlabel('x'), ylabel('sin(x)'), title('Linear Interpolation')

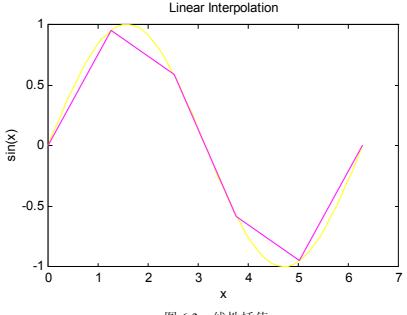


图 6.3 线性插值

图 6.3 是 sine 函数的两个图,一个在数据点之间用 60 个点,它比另一个只用 6 个点更光滑和更精确.

如曲线拟合一样,插值要作决策. 根据所作的假设,有多种插值. 而且,可以在一维以上空间中进行插值. 即如果有反映两个变量函数的插值, z=f(x, y), 那么就可在 x 之间和在 y 之间,找出 z 的中间值进行插值. MATLAB 在一维函数 interp1 和在二维函数 interp2 中,提供了许多的插值选择. 其中的每个函数将在下面阐述.

为了说明一维插值, 考虑下列问题, 12 小时内, 一小时测量一次室外温度. 数据存储在两个 MATLAB 变量中.

hours=1:12; % index for hour data was recorded temps=[5 8 9 15 25 29 31 30 22 25 27 24]; % recorded temperatures plot(hours, temps, hours, temps,'+') % view temperatures title('Temperature') xlabel('Hour'), ylabel('Degrees Celsius')

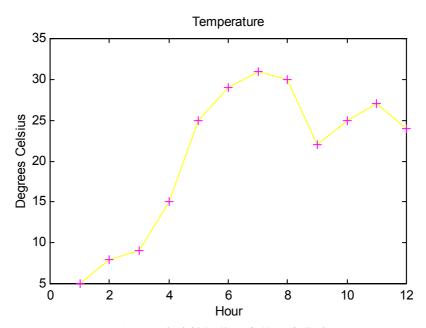


图 6.4 在线性插值下室外温度曲线

正如图 6.4 看到的, MATLAB 画出了数据点线性插值的直线. 为了计算在任意给定时间的温度, 人们可试着对可视的图作解释. 另外一种方法, 可用函数 interp1.

```
t=interp1(hours, temps, 9.3)  % estimate temperature at hour=9.3

t = 22.9000

t=interp1(hours, temps, 4.7)  % estimate temperature at hour=4.7

t = 22

t=interp1(hours, temps, [3.2 6.5 5.1 11.7])  % find temp at many points!

t = 10.2000
    30.0000
    30.9000
    24.9000
```

interp1 的缺省用法是由 interp1(x, y, xo)来描述,这里 x 是独立变量(横坐标),y 是应变量(纵坐标),xo 是进行插值的一个数值数组.另外,该缺省的使用假定为线性插值.

若不采用直线连接数据点,我们可采用某些更光滑的曲线来拟合数据点.最常用的方法是用一个3阶多项式,即3次多项式,来对相继数据点之间的各段建模,每个3次多项式的头两个导数与该数据点相一致.这种类型的插值被称为3次样条或简称为样条.函数 interp1 也能执行3次样条插值.

```
t=interp1(hours, temps, 9.3, 'spline') % estimate temperature at hour=9.3
t =
21.8577
t=interp1(hours, temps, 4.7, 'spline') % estimate temperature at hour=4.7
t =
22.3143
t=interp1(hours, temps, [3.2 6.5 5.1 11.7], 'spline')
t =
9.6734
```

30.0427

31.1755

25.3820

注意, 样条插值得到的结果, 与上面所示的线性插值的结果不同. 因为插值是一个估计或猜测的过程, 其意义在于, 应用不同的估计规则导致不同的结果.

一个最常用的样条插值是对数据平滑. 例如,

h=1:0.1:12; % estimate temperature every 1/10 hour

t=interp1(hours, temps, h, 'spline');

title('Springfield Temperature')

xlabel('Hour'), ylabel('Degrees Celsius')

在图 6.5 中, 虚线是线性插值, 实线是平滑的样条插值, 标有'+'的是原始数据. 如要求在时间轴上有更细的分辨率, 并使用样条插值, 我们有一个更平滑、但不一定更精确地对温度的估计. 尤其应注意, 在数据点, 样条解的斜率不突然改变. 作为这个平滑插值的回报, 3 次样条插值要求更大量的计算, 因为必须找到 3 次多项式以描述给定数据之间的特征.

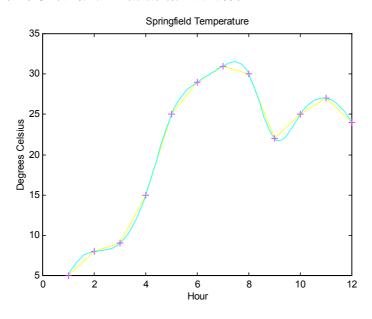


图 6.5 在不同插值下室外温度曲线

6.3 小结

下面的表 6.1 总结了在 MATLAB 中所具有的曲线拟合和插值函数.

表 6.1

	曲线拟合和插值函数
polyfit(x, y, n)	对描述 n 阶多项式 y=f(x)的数据, 进行最小二乘曲线拟合
interp1(x, y, xo)	1 维线性插值
interp1(x, y, xo, 'spline')	1 维 3 次样条插值
interp1(x, y, xo, 'cubic ')	1 维 3 次插值
interp2(x, y, Z, xi, yi)	2 维线性插值
interp2(x, y, Z, xi, yi, 'cubic ')	2 维 3 次插值
interp2(x, y, Z, xi, yi, 'nearest ')	2 维最近邻插值

§ 1.8 求解非线性规划

```
非线性规划的标准型为:
        \min F(X)
        s.t.G(X) \leq 0
        VLB≤X≤VUB
其中 X \in \text{En}, G(X) \in \text{Em}.
    用 Matlab 求解, 分两步:
    1. 首先建立函数形式的 M 文件 fun.m:
         function[f,g]=fun(X);
         f=F(X);
         g=[G1(X); ...; Gm(X)]
    2. 命令的基本格式有三个:
    (1)X=constr('fun', x0)
    (2)X=constr('fun', x0, options)
    (3)X=constr('fun', x0, options, VLB, VUB)
其中 x0, X 为迭代初值, options 是参数说明向量.
    例 求解以下模型
    \min f(x) = e^{x_1} (4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)
             x_1 + x_2 = 0
    s.t. \left\{ 1.5 + x_1 x_2 - x_1 - x_2 \le 0 \right\}
            -x_1x_2 - 10 \le 0
    解: (1)先建立 M-文件 fun1.m:
            function[f,g]=fun(X);
            f=\exp(x(1)) * (4 * x(1)^2+2 * x(2)^2+4 * x(1) * x(2)+2 * x(2)+1);
            g=[x(1)+x(2); 1.5+x(1) * x(2)-x(1)-x(2); -x(1) * x(2)-10];
   (2)再输入以下命令:
         x0=[-2,2];
         options(13)=1; (说明 g 中第一个约束为等式约束)
          [x, options] = constr('fun1', x0, options)
          options(8)
                             (表示输出在最后极值点的函数值)
   (3)运算结果:
        x = -1.2247
            1.2247
        ans = 1.8951
```

第二节 Maple

现在的计算机不仅可以进行数值计算,还可以进行符号运算.数值计算就是处理的数据和处理后的结果都是数值,而符号运算则是处理的数据和处理后的结果都是符号.所谓符号,可以是字母、公式,也可以是数字.符号计算实际上就是数学演算、数学推理和数学证明.专门用来完成这些演

算、推理和证明的软件,通常称为数学软件. 当今比较成熟的数学软件有 MatLab、Maple 和 mathematic. MatLab 主要是进行矩阵的运算,具有很强的数值运算能力;而 Maple 和 mathematic 则擅长符号运算,例如公式的推导、变形、方程的符号解、微积分运算等等;它们各有自己的优点,并列为三大数学软件.

§ 1.1 初识 Maple

1. Maple 简介

Maple 是一个优秀的国际流行的数学软件系统,它提供了广泛的数学计算功能,主要包括三个方面:符号演算、二维和三维图形描绘以及数值计算. Maple 能解决如下问题:

初等数学中的有理数运算(精确计算和大有理数计算),代数式运算,初等函数变换,映射与集合运算,方程求根等.

复数运算,向量代数,平面和空间解析几何,线性代数中矩阵的各种运算和变换,线性方程组求解.

微积分中的函数的极限,导数,积分,泰勒级数、傅立叶级数展开,以及数列与级数求和,求积等.

微分方程的解析解, 数值解以及通过方向场图形得出近似解.

此外, Maple 还提供了解决诸如图论、数论、群论、组合数学、概率统计、特殊函数(运算微积)和其他应用数学(如金融数学)方面的问题的软件包. 总之, 凡是能用公式解决的问题, Maple 都能解决; 它能有效地辅助人们在数学计算方面的繁琐事务, 进而去从事更加智能化和深层次的问题的研究.

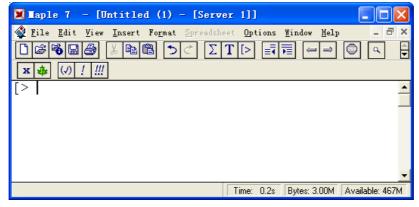
Maple 是加拿大 Waterloo 大学符号计算小组于 1980 年开始研制, 经过多年的应用和开发, 至今发展到 Maple9. 由于枫树是加拿大的国树, Maple 的开发者为了表示自己的爱国之情, 所以把枫叶作为 Maple 的图标和标志.

Maple 系统包含有三部分: 用户界面 Iris, 基本部分(内核)Kernel 和外部库 Liberary:

Iris. 语法分析(对用户的输入进行分析), 主要功能: 表达式显示, 图形处理, 特殊用户界面.

Kernel. 翻译器(将用户输入翻译成机器代码), 主要功能: 内存管理, 整数域, 有理数域, 实数域, 扩展有理数域的计算和处理.

Liberary. 外部库函数,包括应用软件包和在线帮助.



Maple 操作界面

用户界面 Iris 和基本部分 Kernel 是 Maple 的主体,由 C++语言编写而成,用户启用使用 Maple 时就将它们调入内存,直接可以执行.外部库 Liberary 的库函数由 Maple 语言编写而成,在用户需要时才调入内存.

Maple 还提供了方便的帮助系统, 只要键入相关的关键字, 按下 F1 键, 或者在工作窗口中输入"?

关键字"再按 Enter, 就可以得到详细的说明和例子; 通常, 每一个在线帮助系统的画面的最下方都有有一些链接(links), 提供了一些其它相关的主题. 如果要查询哪个主题, 按一下该链接即可显示相关性的主题的画面.

2. 第一次使用 Maple

启动 Maple 之后, Maple 会自动开启一个新的"工作窗口". 每一个工作窗口即为一个文件, 而 Maple 用一种特殊的文件格式来存储工作窗口中的内容, 其文件名后缀为.mws (Maple WorkSheet). 假设要求解方程式 $x^2+ax-b=0$ 的根, 可以在工作窗口中键入:

solve($x^2+a*x-b=0,\{x\}$);

再按 Enter 键来执行这个简单的运算, 此时您会得到如下的画面:

注意事项:

- 1. 上图中的提示符号(>)为 Maple 自动产生. 当您输入算式时, 一定不要把提示符号也敲进去, 否则会有错误的消息产生.
- 2. Maple 使用":="作为赋值号而不是"=".
- 3. Maple 表达式的结束符是";"和":",以";"结束时 Maple 显示结果,而以":"结束则不显示结果. 命令行最左边的"["所框的范围代表一个工作组, Maple 每收到一个执行指令(按 Enter 键或点击

₩ 按钮)就执行当前光标在的工作内的所有命令. 也就是说, 在同一个工作组内的命令总是同时执行的. 如果想一次执行一个文件内的所有命令, 可以点击 按钮.

通常 Maple 在执行完最后一条命令后在后面产生一个空的命令输入行. 如果想人为增加, 可用 鼠标点击工具栏上的 按钮.

4. 因为按 Enter 是执行命令, 想强迫换行必须按 Shift+Enter.

当执行的语句很多以后,前面变量的值会影响到后面的语句而得到不是您所期望的结果,可以通过执行 restart:或点击 按钮来清除所有用户定义变量的值.

§ 1.2 初等数学

1. 阶乘!和 factorial

格式: n! 或 factorial(n) ——求 n 的阶乘.

2. 绝对值 abs

格式: abs(x) ——求 x 的绝对值.

3. 质数测试 isprime 及求质数 nextprime

格式: isprime(n) ——测试整数 n 是否是质数. nextprime(n) ——求出比 n 大的最小质数.

4. 最大数 max 和最小数 min

格式: $\max(x1, x2, ...)$ ——相当于 $\max\{x1, x2, ...\}$, 即求 x1, x2, ...的最大数. $\min(x1, x2, ...)$ ——求 x1, x2, ...的最小数.

5. 随机数 rand

格式: rand()——返回 12 位的非负随机整数. rand(a..b)——生成一个函数,该函可以产生区间[a,b]内的随机整数.

6. 多项式插值函数 interp

格式: interp(x, y, v) ——函数 interp(x, y, v)根据下列独立点(x[1],y[1]), (x[2],y[2]), ... (x[n],y[n]) 计算小于或等于 n 次的插值多项式.

例

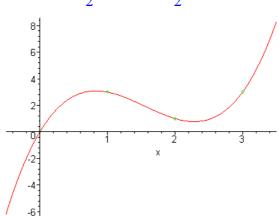
with(plots):

p:=interp([0,1,2,3],[0,3,1,3],x);

poin:=pointplot({[0,0],[1,3],[2,1],[3,3]},color=green): #作数据点图, 只把数据用 poin 保存, 暂不显示

line:=plot(p,x=-0.5..3.5): #作多项式 p 的图形, 暂不显示 display(poin,line); #显示图形

$$p := \frac{3}{2}x^3 - 7x^2 + \frac{17}{2}x$$



7. 排序 sort

格式: 1. sort(L) 若 L 是表, 则 sort(L)将 L 的元素按升序排序; 若 L 是代数表达, 则 sort(L)将 L 按降幂排列;

2. sort(L,F) F 是两个参数的布尔过程, 按 F 指定方法进行降幂排列.

8. 求和 sum、add 与求积 product、mul

格式: sum(f, k=m..n) 计算 $\sum_{k=m}^{n} f$.

add(f, k=m..n) 计算
$$\sum_{k=m}^{n} f$$
.

product(f, k=m..n) 计算
$$\prod_{k=m}^{n} f$$
.

$$\text{mul}(\mathbf{f}, \mathbf{k}=\mathbf{m}..\mathbf{n})$$
 计算 $\prod_{k=m}^{n} f$.

9. 几个用于表达式化简的函数——simplify、factor、combine、expand

格式: simplify(expr) 对表达式 expr 进行化简, 化简一词在数学上是含糊不清的, 但 Maple 常常能得到令人满意的结果. 如果用 simplify 化简仍达不到要求, 可以考虑结合其它几个化简函数一起使用.

factor(expr) 对表达式 expr 进行因式分解.

combine(expr) 对表达式 expr 进行合并.

expand(expr) 对表达式 expr 进行展开.

这些函数是非常有用的,如果某个命令求得的结果非常的冗长,您应该尝试用这些命令对它进行化简.这些命令的用法在此并没有完全列出,详细请查阅 Maple 系统自带的帮助.

例

simplify($\cos(x)^5 + \sin(x)^4 + 2*\cos(x)^2 - 2*\sin(x)^2 - \cos(2*x)$);

$$\cos(x)^5 + \cos(x)^4$$

combine(exp(sin(a)*cos(b))*exp(cos(a)*sin(b)));

$$e^{\sin(a+b)}$$

factor(x^3+y^3);

$$(x+y)(x^2-xy+y^2)$$

expand(sin(x+y));

$$\sin(x)\cos(y) + \cos(x)\sin(y)$$

- 10. 求表达式值 eval、evalf、evalm、evalb、value
- (1) 格式: eval(expr) 求出表达式 expr 的最终值. eval(expr, x=a) 求出表达式 expr 在 x=a 处的值.
- (2) 格式: evalf(expr) 以默认的精度求出表达式 expr 的近似值. evalf[n](expr) 求出表达式 expr 近似值, 并且精确到小数点后面 n 位.
- (3) 格式: evalm(matrix expr) 求出矩阵 表达式 matrix expr 的值.

 Maple 为了提高运行速度,有些变量并不求出最终的值,而是以变量的形式参与,
 矩阵运算尤其如此.为了求出矩阵的值,使用 evalm.
- (4) 格式: evalb(Boolean expr) 求出布尔表达式 matrix expr 的值 true 或 false, 如果无法判断返回原表达式. 布尔表达式由关系符>(大于)、>=(大于等于)、=(等于)、<=(小于等于)、<(小于)、<>(不等于)及逻辑操作符 and(与)、or(或)、not(非)将变量或表达式连接而成.
- (5) 格式: value(expr) 求出表达式 expr 的值.

 Maple 的极限、求导、积分、求和、求积等运算中,都有一种命令是只写出求表达式而没有真正计算,这些命令通常以大写字母开头,例如极限 Limit,求导 Diff,积分 Int,求和 Sum,求积 Product,如果要求出其值,一是使用小写字母开头的命令,再就是使用 value().

§1.3 方程求解

1. 解方程 solve

格式: solve(eqns, vars)

说明: solve(eqns, vars)用来求解方程或方程组 eqns, vars 指出未知数.

eqns 可以是线性方程、非线性方程,可以是多项式、三解函数、指数和绝对值方程,甚至可以是不等式.

如果方程不存在显式解或显式解过于复杂,则返回RootOf形式的解,RootOf(expr)的意思是expr的解.

解绝对值方程,函数 solve 假定 abs 的参数是实数值.

有时, Maple 不能找到解, 并不表明解不存在, 只是 Maple 不能找到它. 这时可以尝试用 fsolve 求它的近似解.

例

eqns := {u+v+w=1,
$$3*u+v=3$$
, $u-2*v-w=0$ }; #解三元线性方程组
eqns := { $u+v+w=1$, $3u+v=3$, $u-2v-w=0$ }

sols := solve(eqns);

sols :=
$$\{w = \frac{-2}{5}, u = \frac{4}{5}, v = \frac{3}{5}\}$$

eq := x^4-5*x^2+6*x=2; #求解高次多项式的根

$$eq := x^4 - 5x^2 + 6x = 2$$

solve(eq,x);

1, 1,
$$-1 + \sqrt{3}$$
, $-1 - \sqrt{3}$

solve(x-sin(x), x); #求解三角函数议程

0

sovle(x^2+x>5, x); #求解不等式

sovle(
$$5 < x^2 + x, x$$
)

solve(abs(x)>=5, x); #求解绝对值不等式

RealRange(
$$5, \infty$$
), RealRange($-\infty, -5$)

2. 求方程近似解 fsolve

格式: fsolve(eqns, vars)

说明: 对一般的方程, fsolve 计算出一个实根, 对多项式计算出所有实根(非复数).

有时用 solve 求方程的解的结果非常冗长或者是求不出(但解存在), 用 fsolve 往往能得到满意的结果.

例

solve(tan(sin(x))=1, x);

$$\arcsin\left(\frac{1}{4}\pi\right)$$

fsolve(tan(sin(x))=1, x);

.9033391108

poly :=
$$23*x^5 + 105*x^4 - 10*x^2 + 17*x$$
: solve(poly, x);

0, RootOf(23
$$Z^4 + 105 Z^3 - 10 Z + 17$$
, index = 1),

RootOf(23
$$Z^4 + 105 Z^3 - 10 Z + 17$$
, index = 2),

RootOf(23
$$Z^4 + 105 Z^3 - 10 Z + 17$$
, index = 3),

RootOf(23
$$Z^4 + 105 Z^3 - 10 Z + 17$$
, index = 4)

fsolve(poly, x, -1..1);

-4.536168981, -.6371813185, 0.

3. 方程整数解 isolve

格式: isolve(eqns, vars)

说明: isolve(eqns, vars)求出方程 eqns 的整数解.

返回解中若含有变量_N1,_N2, 其含义是通用变量.

如果返回 NULL, 则说明 Maple 找不到解, 但并没有保证真的无解.

例

isolve(3*x-4*y=7);

$$\{x = 5 + 4 \ Z1, y = 2 + 3 \ Z1\}$$

4. 解模方程 msovle

格式: msolve(eqns, m)

msolve(eqns, vars, m)

说明: msolve(eqns, m)求方程的所有求知数关于模 m 的整数解. 解里面含有通用变量_NN1, ...,

NNn 表示整数参数.

msolve(eqns, vars, m)求方程指定求知数关于模 m 的整数解.

例

msolve({3*x-4*y=1,7*x+y=2},19);

$$\{ y = 11, x = 15 \}$$

 $msolve(x^2=3,5);$

msolve(2^i=3,19);

$$\{i = 13 + 18 \ Z1 \sim \}$$

 $msolve(8^{j}=2,x,17);$

$${j = 3 + 8 x}$$

5. 解递推方程 rsovle

格式: rsolve(eqns, fcns)

说明: rsolve(eqns, fcns)解出递推关系指定的方程 eqns 中的 fcns, 返回 fcns 的通项表达式.

例

 $rsolve({y(n)=n+y(n-1), y(0)=0}, y(n));$

$$(n+1)\left(\frac{1}{2}n+1\right)-n-1$$

simplify(%);

$$\frac{1}{2}n^2 + \frac{1}{2}n$$

 $rsolve({f(n) = f(n-1) + f(n-2), f(0)=1, f(1)=1},{f(n)});$

$$\left\{ f(n) = \frac{2}{5} \frac{\sqrt{5} \left(2 \frac{1}{-1 + \sqrt{5}} \right)^n}{-1 + \sqrt{5}} + \frac{\frac{2}{5} \sqrt{5} \left(-2 \frac{1}{\sqrt{5} + 1} \right)^n}{\sqrt{5} + 1} \right\}$$

§ 1.4 线性代数(矩阵运算)

1. 数组定义 array

格式: array(bounds, list)

说明: 参数中至少有范围 bounds 或初值表 list.

Maple 数组的下标可以是负数和零.

array(bounds)生成一个未指定元素值的数组, array(list)直接生成数组, 元素值由 list 指定. array 可以定义多维数组, 形如 A:=array(1..10, 1..10)定义二维数组 A10×10, 多维数组类似.

一维数组的 list 形如[a1,a2,...,an], 二维数组的 list 形如[[a11,a12,...,a1n],[a21,a22,...,a2n], ..., [am1,am2,...,amn]], 以此类推.

Maple 以索引来表示数组.

Maple 用 A[1,1]来取得二维数组 A 的元素, 而不是 A[1][1].

C语言中限定同一个数组中的元素必须具有相同的数据类型, Maple 并没有此限制.

例

A:= array(-1..1, -1..1);

$$A := array(-1 ... 1, -1 ... 1, [])$$

A[-1,0]:=1;

$$A_{-1,0} := 1$$

 $A[1,1]:=2*x^2+x;$

$$A_{1,1} := 2 x^2 + x$$

lprint(eval(A));

array(-1 .. 1, -1 .. 1,
$$[(-1, 0)=1, (1, 1)=2*x^2+x]$$
)

B:= array(0..1,0..1,[[1,2],[x, x^2]]);

B := array(0 ... 1, 0 ... 1, [

$$(0,0)=1$$

$$(0, 1) = 2$$

$$(1,0) = x$$

$$(1, 1) = x^2$$

1)

B[0,1];

2

2. 向量定义 vector

格式: vector(n, [x1, x2, ..., xn])

vector(n, f)

说明: 参数中至少有向量维数 n 或初值表[x1, x2, ..., xn].

vector(n)定义一个没有确定值的向量.

vector(n, f) = vector(n, [f(1), f(2), ..., f(n)]).

向量是下标从一开始的一维数组,即 vector(n)=array(1..n).

取向量 A 元素用 A[1], A[2]等等.

vector 属于 linalg 软件包.

例

with(linalg): #包含线性代数软件包

Warning, the protected names norm and trace have been redefined and unprotected vector([3,4,5,6,7]);

[3, 4, 5, 6, 7]

V:=vector(4, [1,3,5,7]);

V := [1, 3, 5, 7];

 $W:=vector(4,(i)->i^2);$

W := [1, 4, 9, 16];

A:=array(1..4, [1,3,5,7]);

A := [1, 3, 5, 7];

type(V, array); #向量是一维数组

true

type(A, vector); #下标从 1 开始的数组是向量

true

3. 矩阵定义 matrix

格式: matrix(m, n, [[a11,a12,...,a1n],[a21,a22,...,a2n], ..., [am1,am2,...,amn]]) matrix(m, n, f)

说明: 参数中至少有向量维数 $m \times n$ 或初值表[[a11,a12,...,a1n],[a21,a22,...,a2n], ..., [am1,am2,...,amn]] matrix(m, n)定义一个没有确定值的矩阵.

矩阵是下标从一开始的二维数组,即 matrix(m, n)=array(1..m, 1..n).

取矩阵 A 元素用 A[1, 1], A[1, 2]等等.

matrix 属于 linalg 软件包.

例

with(linalg): #包含线性代数软件包

Warning, the protected names norm and trace have been redefined and unprotected

A:=matrix(3, 3);

$$A := array(1...3, 1...3, [])$$

B:=matrix([[1,2,3], [4,5,6], [7,8,9]]);

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

 $C:=matrix(3, 3, (i,j)->i^2+j^2);$

$$C := \begin{bmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \end{bmatrix}$$

type(C, matrix); #C 是矩阵类型

true

type(C, array); #C 是二维数组

true

4. 矩阵的基本运算

matadd(A,B)或 A+B — 矩阵加法
multiply(A,B)或&*(A,B)或 A&*B — 矩阵乘法
scalarmul(A, λ)或 λ*A — 数乘矩阵
transpose(A) — 转置矩阵 A
inverse(A) — 求 A 的逆矩阵

with(linalg):

A, B:=matrix([[1,2,3], [4,5,6], [7,8,9]]), $matrix(3, 3, (i,j)->i^2+j^2)$;

$$A, B := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \begin{bmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \end{bmatrix}$$

evalm(A+B); #矩阵加法

evalm(&*(A,B)); #矩阵乘法

transpose(A); #矩阵转置

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

inverse(matrix(2,2,[[a,b], [c,d]])); #矩阵求逆

$$\begin{bmatrix} -\frac{d}{-a d+b c} & \frac{b}{-a d+b c} \\ \frac{c}{-a d+b c} & -\frac{a}{-a d+b c} \end{bmatrix}$$

4. 矩阵的初等变换

addrow(A, r1, r2, m) — 将矩阵 A 的 r1 行乘上 m, 再加到 A 的 r2 行上 mulrow(A, r, m) — 将矩阵 A 的 r 行乘上 m swaprow(A, r1, r2) — 将矩阵 A 的 r1 与 r2 行互换 注: 只要将函数名的"row"替换成"col"就可以进行相应的列变换.

5. 解线性联立方程组 AX = B

格式: linsolve(A, B) 求解解线性联立方程组 AX = B, 返回解向量, 属于 linalg 软件包. 例

A:=matrix(3,3,[[1,0,2],[0,4,-3],[3,6,0]];B:=vector(3,[9,1,0]);

6. 矩阵的行列式、特征值和特征向量

det(A) — 计算 A 的行列式值

charpoly(A) — 求矩阵 A 的特征方程

eigenvalues(A) — 求矩阵 A 的特征值

eigenvectors(A) — 求矩阵 A 的特征向量

例 A:=matrix([1,2,3],[1,2,3],[2,5,6]);

对 eigenvectors 矩阵特征向量结果的注释.

§ 1.5 微积分

1. 极限 limit

格式: $\liminf(f(x), x=a, dir)$ — 求 $\lim_{x\to a} f(x)$ dir 指定逼近方向

说明: dir 为逼近方向, 其值可以是 left(左极限)、right(右极限)、real(双方向极限)与 complex(全方向极限). 缺省则以双方向逼近. 逼近的点可以是一个常数、变量或者是 infinity(∞)、-infinity(∞).

complex 用在多元函数

Maple 允许极限为 infinity(∞)

例

limit(sin(x)/x, x=0);

limit(ln(abs(x))/x, x=0); #极限不存在

undefined

1

limit(ln(abs(x))/x, x=0,right); #右极限为无穷大

_ ∝

limit((x+1)/x, x=infinity);

1

limit((sin(x)-sin(a))/(x-a), x=a);

 $\cos(a)$

a:=(n)->1/sqrt(5)*(((1+sqrt(5))/2)^n-((1-sqrt(5))/2)^n); #费波那奇数列通项公式

$$a := n \to \frac{\left(\frac{1}{2} + \frac{1}{2}\sqrt{5}\right)^n - \left(\frac{1}{2} - \frac{1}{2}\sqrt{5}\right)^n}{\sqrt{5}}$$

limit(a(n)/a(n+1), n=infinity);

$$2\frac{1}{1+\sqrt{5}}$$

evalf(%); #黄金分割点

.6180339886

2. 微分 diff、D

格式: diff(f(x),x) — 求
$$\frac{d^n}{dx^n} f(x)$$
,

diff(f(x),x\$n) — 求
$$\frac{d^n}{dx^n} f(x)$$

D(f) — 计算函数 f 的一阶微分

(D@@n)(f) — 计算函数 f 的 n 阶微分和

说明: diff(f(x),x)是对表达式求微分, 而 D(f)中的 f 是函数名, 不是 f(x). diff(f(x1,x2,...),x1,x2,...)和 D[1,2,...](f)用来求多元函数微分 例

/ i

diff(sin(x), x);

cos(x)

diff(sin(x), x\$2);

 $-\sin(x)$

f:=**D**(**sin**);

f:=cos

g:=(D@@2)(sin);

 $g:=-\sin$

f(Pi);

-1

diff('f'(x,y), x, y);

$$\frac{\partial^2}{\partial y \, \partial x} \, \mathbf{f}(x, y)$$

diff(sin(x*y), x, y);

$$-\sin(xy)xy + \cos(xy)$$

 $s:=(x,y)->\sin(x*y);$

$$s := (x, y) \rightarrow \sin(x y)$$

D[1,2](s);

$$(x, y) \rightarrow -\sin(xy) y x + \cos(xy)$$

3. 积分

格式: int(f(x), x) — 求不定积分 $\int f(x)dx$

$$int(f(x), x=a..b)$$
 — 求定积分 $\int_a^b f(x)dx$

Doubleint(
$$f(x,y), x, y$$
) — 求二重不定积分 $\iint f(x,y) dx dy$

Doubleint(f(x,y), x=a..b, y=c..d) — 求二重定积分
$$\int_{c}^{d} \int_{a}^{b} f(x,y) dx dy$$

说明: int(f(x), x)计算不定积分时 Maple 省去了积分常数 C. Doubleint()只求出积分形式,并没有作真正的计算,可用 value()求出值.

int 与 Doubleint 属于 student 软件包.

例

with(student):

 $int(1/(x^3+1), x);$

$$\frac{1}{3}\ln(x+1) - \frac{1}{6}\ln(x^2 - x + 1) + \frac{1}{3}\sqrt{3} \arctan\left(\frac{1}{3}(2x-1)\sqrt{3}\right)$$

 $int(1/(x^3+1), x=0..1);$

$$\frac{1}{3}\ln(2) + \frac{1}{9}\sqrt{3} \pi$$

Doubleint(f(x,y), x, y);

$$\iint f(x,y) \, dx \, dy$$

Doubleint(x*y, x, y);

$$\iint x \ y \ dx \ dy$$

value(%);

$$\frac{1}{4}x^2y^2$$

5. 微分方程求解

格式: dsolve({eqns, init}, f(x)) — 求解微分方程 eqns, init 是初始条件, f(x)指定要求解的函数 pdsolve(eqns, f(x)) — 求解偏微分方程 eqns, f(x)指定要求解的函数

例

eqn:=diff(y(x),x\$2)+2*diff(y(x),x)+1=0;

$$eqn := \left(\frac{\partial^2}{\partial x^2} y(x)\right) + 2\left(\frac{\partial}{\partial x} y(x)\right) + 1 = 0$$

dsolve(eqn);

$$y(x) = -\frac{1}{2}e^{(-2x)}C1 - \frac{1}{2}x + C2$$

dsolve({eqn, y(0)=0, (D@@2)(y)(0)=1}); #解含初值条件的微分方程

$$y(x) = \frac{1}{4} e^{(-2x)} - \frac{1}{2}x - \frac{1}{4}$$

 $PDE:=x*diff(f(x,y),y)-diff(f(x,y),x)=f(x,y)^2*g(x)/h(y);$

$$PDE := x \left(\frac{\partial}{\partial y} f(x, y) \right) - \left(\frac{\partial}{\partial x} f(x, y) \right) = \frac{f(x, y)^2 g(x)}{h(y)}$$

pdsolve(PDE); #解偏微分方程

$$f(x,y) = \frac{1}{\int_{-\frac{1}{2}a^2 + y + \frac{1}{2}x^2}^{x} d_a a + F1\left(y + \frac{1}{2}x^2\right)}$$

§ 1.6 作图

1. 二维曲线作图 plot

格式: plot(f(x), x=a..b) — 作 f(x)的图形, x 在[a, b]内 plot([f1(x), f2(x), ...], x=a..b) 在同一坐标系中作多个函数图形 plot([x(t), y(t), t=a..b]) — t 从 a 到 b 作二维参数图 plot([[x1,y1], [x2,y2], ...], style=point) — 数据点绘图 plot(f(x), x=a..b, options) — 加入选项来更改绘图指令的默认值

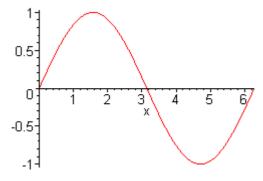
说明: options 的形式如 option=value, 取值如下

- —— axes 指定坐标轴形状: FRAME(框), BOXED(盒子), NORMAL(普通), NONE(无)
- color 指定作图颜色, aquamarine(碧绿), black(黑), blue(蓝), navy(藏青), coral(珊瑚), cyan(墨绿), brown(棕色), gold(金), green(绿), gray(灰白), grey(灰), khaki(土黄), magenta(洋红), maroon(栗色), orange(橘黄), pink(粉红), plum(梅红), red(红), sienna(赫色), tan(黄褐), turquoise(蓝玉), violet(紫), wheat(灰黄), white(白), yellow(黄)
- —— coords 坐标系选择, cartesian(笛卡尔坐标系, 缺省), elliptic(椭圆坐标系), logarithmic(对数坐标系), polar(极坐标)

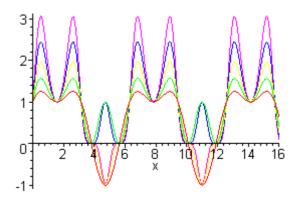
详细的说明请在 Maple 中键入?plot[options]来查询 options 不仅可以用在 plot 函数中, 几乎所有的 Maple 作图函数都可以使用

例

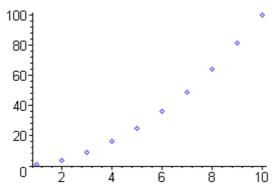
plot(sin(x),x=0...2*Pi);



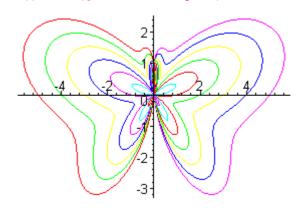
plot($[\sin(x)+\cos(x)^2, (\sin(x)+\cos(x)^2)^2, (\sin(x)+\cos(x)^2)^3, (\sin(x)+\cos(x)^2)^4, (\sin(x)+\cos(x)^2)^5], x=0..16$);



data:=[1,1],[2,4],[3,9],[4,16],[5,25],[6,36],[7,49],[8,64],[9,81],[10,100]: plot([data],style=point,color=blue);



plot([seq(sin(5*t)+n*cos(t),n=-5..5)],t=0..Pi,coords=polar);



2. 三维作图 plot3d

格式: plot3d(f(x,y), x=a..b, y=c..d) — 在 $[a, b] \times [c, d]$ 内作 f(x,y)的图形

plot3d(f(x,y), x=a..b, y=c..d, options) — 加入选项来更改绘图指令的默认值

说明: 用法可参考 plot

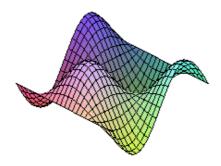
options 取值如下

- —— ambientlight=[r, g, b] 设置光线强度, 数值范围是 0..1
- —— axes 同 plot
- —— color 同 plot
- —— coords 指定坐标系, spherical(球坐标), cylindrical(圆柱坐标)

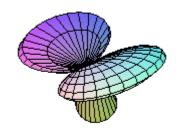
详细的说明请在 Maple 系统中键入?plot3d[options] 查询

例

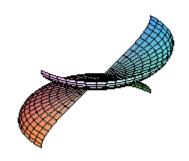
plot3d(sin(x)*cos(y),x=-Pi..Pi,y=-Pi..Pi);



plot3d(sin(t)*sin(p^2),t=0..Pi,p=0..Pi,coords=spherical); #球坐标作图



plot3d(sin(z/2),t=0..2*Pi/2,z=-Pi..Pi,coords=cylindrical); #圆柱坐标作图



3. 显示图形 display

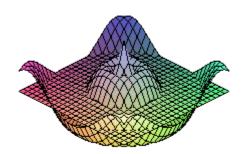
格式: display(g1, g2, ...) — 显示图形 g1, g2, ... 在同一坐标下

说明:如果多个图形无法一次画出,但又想将它们显示在同一坐标下,可以使用 display display 既可以显示二维图,也可以显示三维图,但不能将二维图与三维图同时显示 display 属于 plots 软件包

例

with(plots):

 $g1:=plot3d(2*exp(-sqrt(x^2+y^2)), x=-6..6,y=-6..6):$ $g2:=plot3d(sin(sqrt(x^2+y^2)), x=-6..6,y=-6..6):$ display(g1, g2);



4. Maple 动画

格式: animatecurve(f(x), x=a..b) — 模拟手绘函数 f(x)的过程 animate(f(x,t), x=a..b, t=t1..t2) — 绘制二维平面动画, t 代表时间 animate3d(f(x,y,t), x=a..b, y=c..d, t=t1..t2) — 绘制三维动画, t 代表时间说明: 同属于 plots 软件包.

例

with(plots):

animatecurve([sin(t),cos(t),t=-Pi..Pi]);

animate(sin(x*t), x=0..2*Pi, t=0..8, coords=polar, frames=140);

animate 3 d(cos(x)*sin(y)*sin(t), x =-Pi..Pi, y =-Pi..Pi, t = 0..2*Pi, frames = 36);

(图略)

运行动画函数后 Maple 并没有马上显示动画, 如果想观看动画, 须用鼠标选择图形, 此时在工具

栏上出现播放动画控制按扭 ■ ▶ → ◆ ◆ ◆ ◆ △ 点击即可播放.

5. 图形的保存

Maple 可以把图像转换成各种图形文件格式保存起来. 方法:

用鼠标右键单击图形, 然后选择 Export As, 选择要转换的格式, 再在对话框中选择好保存路径并键入文件名, 点击确定即可.

如果图形是动画,并且保存为 GIF 格式,就可以将 Maple 动画用于网页制作及其它应用了.

§ 1.7 函数与过程

Maple 的函数与过程其实并没有严格的区别,一般来说,函数用"->"箭头操作符来定义,而过程用关键字 proc 来定义. Maple 的函数与过程的调用方法是完全一样的.

a)函数

1. 函数定义

格式: f := (x, y, ...) -> expr —— 定义函数 f(x, y, ...)

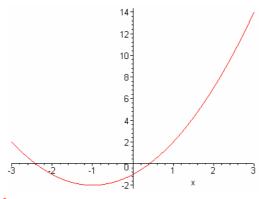
 $f:=(x)->x^2+2*x-1;$

$$f := x \to x^2 + 2x - 1$$

f(sin(x));

$$\sin(x)^2 + 2\sin(x) - 1$$

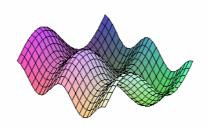
plot(f(x), x=-3..3);



 $g:=(x,y)->\sin(x)+\cos(y)^2;$

$$g := (x, y) \rightarrow \sin(x) + \cos(y)^2$$

plot3d(g(x,y),x=-Pi...Pi,y=-Pi...Pi);



b)过程

1. 过程定义

option o1, o2, ...; — 关于过程的选项, 用于过程记忆表, 操作符, 内部函数组成, 执

行跟踪, 版本说明等

proc body —— 过程体

说明: 由于过程定义较为复杂, 在此将各部分分开解释

(1) 形式参数

格式: p := proc(parem_1::type_1, ..., parem_k::type_1) — parem 变量, type 类型 proc body

end proc;

说明: 调用过程 p(e_1, ..., e_2)的时候, 实际参数表的值从左到右依次传递给形式参数表, 即每一个 e i 的值传给 parem i, 然后检测每个参数的类型. 只要其中之一不匹配, 则返回出错信息.

常用的参数类型 type 有: numeric(数值), float(浮点数), integer(整数), posint(正整数), positive(正数), relacons(实数), complex(复数), even(偶数), odd(奇数), equation(方程), boolean(布尔)等, 更详细的请运行"?type"查看帮助.

参数在过程中不能再赋值.

实际参数的数目可以与形式参数的数目不同, 可以多于但不得少于.

不需要宣告参数类型时, 用形式 proc(x)与 proc(x::anything)相同.

使用变量名 args, 允许用户访问过程内部的实际参数, args[1]取得第一个参数值, args[2]取得第二 参数值, 以此类推. 也就是说, 允许用户不定义形式参数, 但仍然可以访问实际参数.

使用变量 nargs, 可以得到实际参数个数.

因此,使用变量 args 和 nargs,程序变换变得更为灵活

例

f := proc(x, y, z)x+y+z;

end proc;

 $f := \mathbf{proc}(x, y, z) x + y + z$ end \mathbf{proc}

```
f(a, b, c);
                                 a+b+c
   f(1, 2, 3, 4); #实参个数允许多于形参个数, 但此时只有前 3 个有效
   MAX := proc(x::numeric, y::numeric)
           if x>y then x; else y; end if;
   end proc;
      MAX = proc(x::numeric, y::numeric) if y \le x then x else y end if end proc
               #出错, 因为 Pi 不是 numeric 类型
   MAX(Pi, 3);
   Error, invalid input: MAX expects its 1st argument, x, to be of type numeric, but received Pi
   MAX(evalf(Pi), 3);
                  #将 Pi 转化为浮点数后再做比较
                                 3.141592654
   例 3
   #使用变量 args 和 nargs, 不提供形式参数而直接使用实际参数
   #求最大值的函数,参数个数可以是有限个
   MAX := proc()
      local i, m;
      if nargs=0 then return FALL end if;
      m:=args[1];
      for i from 2 to nargs do
         if args[i]>m then m:=args[i] end if;
      end do;
      return m:
   end proc:
   MAX(evalf(Pi), 22/7, 104348/33215);
   MAX(evalf(Pi), 22/7, 104348/33215, 312689/99532, 5419351/1725033);
   (2) 全局变量和局部变量
   变量在过程中要么是局部变量,要么是全局变量. Maple 对于在不同的过程里有相同的变量名的
局部变量, 视为不同的变量. 这样, 在一个过程里的局部变量值的改变, 不影响其它同名的变量值.
   格式: p := proc()
        local v1, v2, ...; —— 定义局部变量
        global g1, g2, ...; —— 定义全局变量
        proc body
        end proc;
   Maple 允许用户不申明变量而直接使用, 那么在过程中未经申明的变量是全局变量还是局部变
量? Maple 自动将下列两种情形视为局部变量, 否则视为全局变量:
   —— 变量出现在赋值号的左边

    一 变量作为下标变量出现在 for 循环中, 或 seq, add, mul 等函数中

    (3) 过程选项
   格式: p := proc()
```

option o1, o2, ...; —— 选项, 用于过程记忆表, 操作符, 内部函数组成, 执行跟踪, 版本说明等 proc body end proc;

说明: 选项有: remember(记忆表), builtin(内部函数组成), system(系统管理), operator(操作符), arrow(箭头符), trace(执行跟踪), Copyright(版本说明).

在此只解释记忆表 remember, 如想了解其它的请自行查阅帮助文档.

使用过程选项 remember, Maple 就在内存中建立一个表(表是类似数组的一种数据结构), 用来存储过程的返回值, 每遇到一个新的实参, 就把这个参数作为表的索引(下标), 把过程的返回值作为表的值. Maple 每执行一次这个过程, 就检查程序是非否存在以前程序执行的结果. 如果程序已经保存这些计算结果, Maple 就直接取回结果而不是再执行同一参数的过程. 这对于执行递归过程特别有用, 使之效率倍增.

使用 op(4, eval(p))可以观察过程 p 的记忆表的结构, 甚至可对记忆表进行各种操作, 操作方法请查阅相关帮助文档.

例

```
#定义费泊那契数列,使用递归定义
```

```
#定义資程が突致力,使用速归定义
fib := proc(n::nonnegint) #参数 n,限定非负整数
    option remember; #使用 remember 选项
    if n<2 then n else fib(n-1) + fib(n-2) end if;
end:
fib(1);

1
fib(4);
```

op(4, eval(fib)); #显示记忆表

table([0 = 0, 1 = 1, 2 = 1, 3 = 2, 4 = 3, 5 = 5, 6 = 8])

8

fib(7);

fib(6);

13

fib(8);

21

op(4, eval(fib)); #显示记忆表

table([0 = 0, 1 = 1, 2 = 1, 3 = 2, 4 = 3, 5 = 5, 6 = 8, 7 = 13, 8 = 21])

fib(4); #当记忆表中有参数为 4 的值时, 过程内的语句不被执行, 而是直接从记忆表中取出结果 3

(4) 过程返回值

格式: RETURN(v_1, v_2, ..., v_n) —— 返回一个或多个值 v_1, v_2, ..., v_n return v_1, v_2, ..., v_n —— 返回一个或多个值 v_1, v_2, ..., v_n ERROR(v_1, v_2, ..., v_n) —— 出错信息 WARNING(msg)—— 警告信息

说明: RETURN 的普遍形式是返回过程执行的最后一条语句, RETURN 与 return 无多大差别. 用 RETURN('procname'(args))可以实现返回过程调用原样. 返回多个值时是返回一个序列, 如果执行 v:=p()而 p()有多个返回值, 那么 v[1], v[2]分别

§ 1.8 流程控制

```
1. 条件判断语句 if
    格式: if cond 1 then
              statement_1;
          elif cond 2 then
              statement 2;
          ... ...
          else
              else_statement;
          end if:
    说明: cond_i 是布尔表达式, statement_i 是语句组.
    当 cond 1 为真(true)时, 执行语句 statement 1, 否则当 cond 2 为真时, 执行语句 statement 2, ...,
当所有的布尔表达式都为假(false)时, 执行语句 else statement.
    elif 和 else 都是可选项,如果不须要的话可以省去.
    一种简化的 if 形式:
    'if (cond, x, y) — 如果 cond 为真(true), 返回 x, 否则返回 y. 注意这里的 if 必须用反单引号(在
键盘左上角 1 旁边的键)括起来.
    ABS := (x) -> if (x>0) then x; else -x; end if; #绝对值函数
      ABS := \mathbf{proc}(x) option operator, arrow, if 0 \le x then x else -x end if end proc
    ABS(3);
                                            3
    ABS(-3);
                                            3
    ABS(a); #当参数为符号时出错
Error, (in ABS) cannot evaluate boolean: -a < 0
    ABS := (x) \rightarrow
                                      #改进的绝对值函数
    if not type(x, numeric) then
         RETURN('procname'(args))(x>0)
      else
         if (x>0) then x; else -x; end if;
ABS := \mathbf{proc}(x) options operator, arrow; if not type(x,numeric) then RETURN(('procname')(args))(0 < x) else
                           if 0 \le x then x else -x end if end if end proc
    ABS(-4);
    ABS(a);
                                          ABS(a)
    2. 循环语句
    格式: for var from v0 by step to v1 while cond do
```

ឲ

省为 infinity(∞), while 缺省为 true.

```
for i from 10 by 5 to 30 do print(i) end do;
                                            10
                                            15
                                            20
                                            25
                                            30
i:=5;
                                            i:=5
while i<20 do i:=i+5; end do;
                                           i = 10
                                           i = 15
                                           i = 20
#一个求<=n 的所有质数的例子
prim := proc(n)
  local p, i;
   p[1]:=2;
   for i do
     p[i+1]:=nextprime(p[i]);
     if nextprime(p[i+1])>n then break; end if;
   end do:
   return op(p);
end proc:
p := prim(13);
                 p := table([1 = 2, 2 = 3, 3 = 5, 4 = 7, 5 = 11, 6 = 13])
p[1];
                                             2
p[6];
                                             13
```

当遇到了 break 命令, 退出当前循环.

第三节 LINDO与LINGO

§1.1 LINDO与LINGO简介

LINDO 是 Linear INteractive and Discrete Optimizer 字首的缩写形式,是由 Linus Schrage 于 1986年开发的优化计算软件包,可以用来求解线性规划 (LP——Linear Programming),整数规划 (IP——Integer Programming)和二次规划 (QP——Quadratic Programming)问题. LINDO 易于规划问

题的输入、求解和分析,程序执行速度很快. LINDO 学生版最多可求解多达 200 个变量和 100 个约束的规划问题.

LINGO 可用于求解线性规划和整数规划问题.

LINGO 与 LINDO 不同的是, LINGO 包含了内置的建模语言,允许以简练、直观的方式描述所需求解的问题,模型中所需的数据可以以一定格式保存在列表(List)和表格(Table)中,也可以保存在独立的文件中.

§ 1.2 用 LINDO 求解线性规划(LP)问题

初试 LINDO

LINDO 的求解机制: LINDO 的求解过程采用单纯形法,一般是首先寻求一个可行解,在有可行解情况下再寻求最优解. 用 LINDO 求解一个 LP 问题会得到如下的几种结果:

不可行(No feasible solution) 或 可行(Feasible)

可行时又可分为:有最优解(Optimal Solution)和解无界(Unbounded Solution)两种情况.由于在实际问题中,不太可能出现最大利润无上限的情形,所以使用者应检查是否少了一个约束或有其它印刷错误.

让我们来解如下 LP 问题:

max z=2x+3ys.t. $4x+3y \le 10$ $3x+5y \le 12$ $x,y \ge 0$

由于 LINDO 中已假设所有的变量都是非负的, 所以非负约束可不必再输入到计算机中; LINDO 也不区分变量中的大小写字符(实际上任何小写字符将被转换为大写字符); 约束条件中的"<="及 ">="可用"<"及">"代替. 上面问题用键盘输入如下:

MAX 2X + 3Y

ST

4X + 3Y < 10

3x + 5Y < 12

END

LINDO 中一般称上面这种问题实例(INSTANCE)为模型(MODEL). 以后涉及该模型时,目标函数为第一行,两个约束条件分别为第二、三行. 点击 求解模型, 结果如下:

LP OPTIMUM FOUND AT STEP 2

OBJECTIVE FUNCTION VALUE

1) 7.454545

,		
VARIABLE	VALUE	REDUCED COST
X	1.272727	0.000000
Y	1.636364	0.000000
ROW	SLACK OR SURPLUS	DUAL PRICES
2)	0.000000	0.090909
3)	0.000000	0.545455

NO. ITERATIONS=

计算结果表明:

"LP OPTIMUM FOUND AT STEP 2"表示单纯形法在两次迭代(旋转)后得到最优解.

"OBJECTIVE FUNCTION VALUE 1) 7.4545450 "表示最优目标值为 7.4545450.

"VALUE"给出最优解中各变量(VARIABLE)的值: X =1.272727, Y =1.636364.

"REDUCED COST"给出最优单纯形表中第 0 行中变量的系数 (max 型问题). 其中基变量的 reduced cost 值应为 0, 对于非基变量,相应的 reduced cost 值表示当该非基变量增加一个单位时目标函数减少的量. 本例中此值均为 0.

"SLACK OR SURPLUS"给出松驰变量的值:第 2、3 行松驰变量均为 0,说明对于最优解来讲,两个约束(第 2、3 行)均取等号.

"DUAL PRICES" 给出对偶价格的值: 第 2、3 行对偶价格分别为 .090909, .545455.

"NO. ITERATIONS=2"表示用单纯形法进行了两次迭代(旋转).

一个问题解答之后, LINDO 会询问是否需要做灵敏性分析(DO RANGE (SENSITIVITY) ANALYSIS?) 如果你不需要, 你应回答"N"(NO).

存储模型

下面即是一个具体应用的例子: (可参照上述使用步骤)

首先输入问题:

MAX 60 DESKS + 30 TABLES + 20 CHAIRS

SUBJECT TO

- 2) 8 DESKS + 6 TABLES + CHAIRS <= 48
- 3) 4 DESKS + 2 TABLES + 1.5 CHAIRS <= 20
- 4) 2 DESKS + 1.5 TABLES + 0.5 CHAIRS <= 8
- 5) TABLES <= 5

END

问题求解, 要求做敏感性分析

NO. ITERATIONS=

LP OPTIMUM FOUND AT STEP

OBJECTIVE FUNCTION VALUE

1) 280.0000 VARIABLE VALUE REDUCED COST DESKS 2.000000 0.000000TABLES 0.000000 5.000000 8.000000 0.000000 CHAIRS ROW SLACK OR SURPLUS **DUAL PRICES** 2) 24.000000 0.000000 3) 0.00000010.000000 10.000000 4) 0.000000 5.000000 0.000000 5)

RANGES IN WHICH THE BASIS IS UNCHANGED:

OBJ COEFFICIENT RANGES

VARIABLE	CURRENT	ALLOWABLE	ALLOWABLE
	COEF	INCREASE	DECREASE
DESKS	60.000000	20.000000	4.000000
TABLES	30.000000	5.000000	INFINITY
CHAIRS	20.000000	2.500000	5.000000
	RIGH	ITHAND SIDE RANGI	ES
$\mathbf{p} \mathbf{O} \mathbf{W}$	CUDDENT	ALLOWADIE	ALLOWADIE

ROW CURRENT ALLOWABLE ALLOWABLE

	RHS	INCREASE	DECREASE
2	48.000000	INFINITY	24.000000
3	20.000000	4.000000	4.000000
4	8.000000	2.000000	1.333333
5	5.000000	INFINITY	5.000000

仍以上面的问题 DAKOTA 为例, 下面给出其结果的一般注释:

"LP OPTIMUM FOUND AT STEP2"表示 LINDO 在(用单纯形法)两次迭代或旋转后得到最优解.

"OBJECTIVE FUNCTION VALUE 280.000000"表示最优目标值为 280.

"VALUE"给出最优解中各变量的值. 例 Dakota 问题中需造 2 个(书桌)desks, 0 个(桌子)tables, 和 8 个(椅子) chairs.

"SLACK OR SURPLUS"给出松驰变量的值. 上例中

- s1= 第 2 行松驰变量 =24
- s2= 第3行松驰变量 =0
- s3= 第 4 行松驰变量 =0
- s4= 第 5 行松驰变量 =5

"REDUCED COST"出最优单纯形表中第 0 行中变量的系数(max 型问题). 其中基变量的 reduced cost 值应为 0, 对于非基变量, 相应的 reduced cost 值表示当增加一个单位时目标函数减少的量.

敏感性分析

使用 LINDO 时,结果输出中会提供敏感性分析.这一信息一般包含于两个标题之下,其一是 REDUCED COSTS,另一个是 DUAL PRICES.它们分别表示了当变量或约束条件有微小变动时,目标函数的变化率.

在输出结果中对应于每个变量都有一个 REDUCED COST, 若其数值为 x, 表示对应的变量为零时, 若增加 1 个单位, 目标函数将减少 x 个单位.

输出结果中对应于每一个约束也都有一个 DUAL PRICE. 若其数值为 x,表示对应约束中不等式 右端项若减少 1 个单位,目标函数将增加 x 个单位.

如果 REDUCED COST 或 DUAL PRICE 的值为 0,表示微小扰动不影响目标函数.

有时, 通过分析 DUAL PRICE, 也可对产生不可行问题的原因有所了解.

注意事项:

- 1) LINDO 中已假定所有变量非负. 变量名不能超过 8 个字符.
- 2) 如要输入 <= 或 >= 型约束, 相应以< 或 >代替即可.
- 3) LINDO 不允许变量出现在一个约束条件的右端.
- 4) 目标函数及各约束条件之间一定要有空格分开.
- 5) 一般 LINDO 中不能接受括号()和逗号",",例:400(X1+X2)需写为 400X1+400X2; 10,000 需写为 10000.
- 6) 14)LINDO 将目标函数所在行作为第一行,从第二行起为约束条件. 行号自动产生,也可以人为定义行号或行名. 行名和变量名一样,不能超过 8 个字符.
- 7) 15)数值均衡化及其它考虑. LINDO 不能将 LP 中的矩阵进行数值均衡化. 为了避免数值问题, 使用者应自己对矩阵的行列进行均衡化. 一个原则是, 系数矩阵中非零元的绝对值不能大于100,000 或者小于 0.0001. 如果 LINDO 觉得矩阵元素之间很不均衡, 将会给出警告.
- 8) 量纲分析与一般错误的避免.

LINDO 部分常命令中文注释如下:

MAX/MIN 命令: 用于输入一个包含目标函数,约束条件在内的 LP 模型.

输入程序如下:输入"MAX"(或"MIN"),继之以自然格式的目标函数作为第一行;再输入"SUBJECT TO"(可简写为"ST"),后面跟约束条件行.最后,输入"END"回到命令状态模式.以后只需

给出"GO"命令即可开始优化求解过程.

其中,变量名可以由 1—8 个字母或数字型的字符构成,且第一个字符必须是字母.变量系数不能是指数型,例如: .258E+29 形式的系数是不允许的. 关键词 ("MAX","ST","END"...)及各行之间必须用一个或多个空格分隔开. 空格可以出现在一行之中,但不能出现在变量名中. 一个回车符等价于一个空格.

下面是同一问题的两种合法的输入方式:

1) MIN 2X+3Y

SUBJECT TO -5X-2Z<=10

+10X - Y > 5

END

2) MIN 2X + 3Y

ST -5X-2Z<10

10X - Y > +5

END

其它一些有用的符号有:

算术运算符 逻辑运算符 关系运算符 顺序运算符

 $+ - / * ^ AND.$ > < = #

LOG() EXP() .OR. ABS() .NOT.

GIN 命令:

GIN 命令可将问题模型中的变量标为 0/1 型. 第一种格式为"GIN n", 其中 n 是整型变量的个数. LINDO 要求整型变量应放在问题模型的最前面. 第二种格式为 "GIN var-id", 其中"var-id"是变量名.

§ 1.3 用 LINDO 求解整数规划和二次规划问题

3.1 整数规划(IP)

LINDO 可用于求解单纯的或混合型的整数规划(IP)问题. 但目前尚无相应完善的敏感性分析理论. IP 问题的输入与 LP 问题类似, 但在 END 标志后需定义整型变量.

0/1 型的变量可由 INTEGER(可简写为 INT)命令来标识:

INTEGER vname 或 INTEGER n

前者只将变量 vname 标识为 0/1 型,后者将当前模型中前 n 个变量标识为 0/1 型.模型中变量顺序由输入决定,该顺序可由输出结果中查证.

具体演示如下:

MAX 4TOM +3DICK +2HARRY

ST

2.5TOM +3.1HARRRY < 5

.2TOM + .7DICK + .4HARRY < 1

END

:INT TOM

:INT DICK

:INT HARRY

求解结果:

LP OPTIMUM FOUND AT STEP

OBJECTIVE VALUE = 7.71428585

NEW INTEGER SOLUTION OF 7.00000000 AT BRANCH 0 PIVOT 3 RE-INSTALLING BEST SOLUTION...

OBJECTIVE FUNCTION VALUE

1)	7.000000

-)	7.000000	
VARIABLE	VALUE	REDUCED COST
TOM	1.000000	-4.000000
DICK	1.000000	-3.000000
HARRY	0.000000	-2.000000
HARRRY	0.000000	0.000000
ROW	SLACK OR SURPLUS	DUAL PRICES
2)	2.500000	0.000000
3)	0.100000	0.000000

NO. ITERATIONS= 3

BRANCHES= 0 DETERM.= 1.000E 0

注:整数规划中变量不一定限于 0/1 型,因此 LINDO 中有一个比 INT 弱些的命令 GIN,可将变量仅限为整数型,而其使用方式及格式与 INT 命令相似.

尽管 LINDO 对整数规划问题是很有威力的, 要想有效地使用还是需要一定的技术. 这是因为, 人们很容易将一个本质上很简单的问题列成一个不好的数学表达式, 而一个坏的表达式有可能会导致一个冗长的计算. 当然这时 LINDO 会主动砍去一些过程, 以缩短计算时间.

3.2 二次规划(Quadratic Programming)

Lindo 可用于求解二次规划问题. 例如: 对于

min
$$z=3x^2+y^2-xy+0.4y$$

s.t. $1.2x+0.9y>1.1$
 $x+y=1$
 $y<0.7$

该问题通过在实际约束前增加有关变量的一阶条件,从而转化二次型为线性(互补)型(如下). 这需要我们为每一个实际约束增加一个对偶变量,并要使用 QCP 命令. 第一行(目标函数)只用于给出相应变量的顺序. 对上面的问题,我们将用 RT, ONE 和 UL 作为对偶变量. 问题输入如下:

MIN X+Y+RT+ONE+UL

ST

6X - Y - 1.2RT + ONE > 0.

-X + 2Y - .9RT + ONE + UL > -.4

1.2X + .9Y > 1.1

X + Y = 1

Y < .7

END

求解:

LP OPTIMUM FOUND AT STEP 1 OBJECTIVE FUNCTION VALUE

1.000000

VARIABLE	VALUE	REDUCED COST
X	0.666667	0.000000
Y	0.333333	0.000000
RT	0.000000	1.000000

ONE	0.000000	1.000000
UL	0.000000	1.000000
ROW	SLACK OR SURPLUS	DUAL PRICES
2)	3.666667	0.000000
3)	0.400000	0.000000
4)	0.000000	0.000000
5)	0.000000	-1.000000
6)	0.366667	0.000000
NO. ITERATION	S= 1	

§1.4 LINGO 功能简介

LINGO 软件包的功能之一在于提供了一些复杂的矩阵生成器的例子,更进一步说, LINGO 实际上提供了建立最优化模型的一种语言,有了它,使用者只用键入一行文字也可以建立起成千条约束或目标函数项. 这就使输入较大规模问题的过程得到了简化.

一般地, LINGO 的内部函数有:

普通函数

- @ABS(X)
- @BIN(VAR)
- @SIN(X)
- @COS(X)
- @EXP(X)
- @FPA(I, N)
- @FPL(I, N)
- @FREE(VAR)
- @GIN(VAR)
- @IMPORT(WK1 FILE, RANGE NAME)
- @IN(SET NAME, SET ELEMENT)
- @LGM(N)
- @LOG(X)
- @PBN(P, N, X)
- @PEB(ARL, S)
- @PEL(ARL, S)
- @PHG(POP, GREEN, SAM, X)
- @PFS(ARL, S, C)
- @PPL(ARL, S)
- @PPS(ARL, S)
- @PSN(Z)
- @PSL(Z)
- @SIZE(SET)
- @SMAX(X,Y)
- @SMIN(X,Y)
- @TAN(X)
- @USER(X1, ..., XN)

```
@WARN( 'TEXT', CONDITION)
```

@WRAP(INDEX, SET SIZE)

集合函数

@MAX(SET | COND: EXP)

@MIN(SET | COND: EXP)

@SUM(SET | COND: EXP)

@FOR(SET | COND: EXP)

例: LINGO 中问题的一个典型模型如下: (SAILCO)

MIN
$$\sum_{i=1}^{4} 400RP_i + 450OP_i + 20INV_i$$

ST

 $RP_i < 40, i = 1..4$

$$INV_i = INV_{i-1} + RP_i + OP_i - DEM_i$$
, $TIME_i > 1$, $i = 1..4$
 $INV_1 = 10 + RP_1 + OP_1 - DEM_1$

其中, DEM={40,60,75,25}, TIME={1,2,3,4}.

对应的 LINGO 模型如下:

MODEL:

SETS:

QUARTERS/Q1,Q2,Q3,Q4/:TIME,DEM,RP,OP,INV;

ENDSETS

MIN=@SUM(QUARTERS:400*RP+450*OP+20*INV);

@FOR(QUARTERS(I):RP(I)<40);

@FOR(QUARTERS(I)|TIME(I)#GT#1:

INV(I)=INV(I-1)+RP(I)+OP(I)-DEM(I);;

INV(1)=10+RP(1)+OP(1)-DEM(1);

DATA:

DEM=40,60,75,25;

TIME=1,2,3,4;

ENDDATA

END

从上述 MODEL 中, 我们可看到整个模型分为三大部分:

1)SETS——ENDSETS

其中定义了模型中用到的各个集合,包括变量,数组,变量的特征量等.如上例中定义了四个quarters:Q1,Q2,Q3,Q4. 其中每个quarter都有 TIME,DEM,RP,OP,INV 这样的特征量.一旦这样的定义建立起来,实际上 quarter 的数量可以为 40, 400,或许 4000,它们仍都有 TIME, DEM, RP, OP, INV 这样的特征量.这些量的具体数值可在 DATA——ENDDATA 这部分输入.

2)中间部分:

这部分实际上定义了目标函数,约束条件等.

如上例中: 第 4 行定义了目标函数为 MIN=..., 其中@SUM(quarters: ... 400*RP...)表示对所有的 quarters 计算 400*RP+450*OP+20*INV 并求和. 注意, 如此定义的目标函数与 quarters 的数目是 4, 40, 400. 或 4000 并无具体的关系.

第5行表示对每个quarter, RP不能超过40.

第 6,7 行也是对每个 quarter 中具体约束的定义.

注意:与 LINDO 不同的是变量可以放在约束条件的右端(同时数字也可放在约束条件的左端). 3)DATA——ENDDATA

行 9-12 的作用在于输入必要的数据. 这部分要以 DATA: 开始, 以 ENDDATA 结束. 求解结果如下:

LP OPTIMUM FOUND AT STEP 7

OBJECTIVE VALUE = 78450.0000

, 0 .2 0.0 0 0 0		
VARIABLE	VALUE	REDUCED COST
TIME(Q1)	1.000000	.0000000
TIME(Q2)	2.000000	.0000000
TIME(Q3)	3.000000	.0000000
TIME(Q4)	4.000000	.0000000
DEM(Q1)	40.00000	.0000000
DEM(Q2)	60.00000	.0000000
DEM(Q3)	75.00000	.0000000
DEM(Q4)	25.00000	.0000000
RP(Q1)	40.00000	.0000000
RP(Q2)	40.00000	.0000000
RP(Q3)	40.00000	.0000000
RP(Q4)	25.00000	.0000000
OP(Q1)	.0000000	20.00000
OP(Q2)	10.00000	.0000000
OP(Q3)	35.00000	.0000000
OP(Q4)	.0000000	50.00000
INV(Q1)	10.00000	.0000000
INV(Q2)	.0000000	20.00000
INV(Q3)	.0000000	70.00000
INV(Q4)	.0000000	420.0000
ROW	SLACK OR SURPLUS	DUAL PRICE
1	78450.00	1.000000
2	.0000000	30.00000
3	.0000000	50.00000
4	.0000000	50.00000
5	15.00000	.0000000
6	.0000000	450.0000
7	.0000000	450.0000
8	.0000000	400.0000
9	.0000000	430.0000

LINGO 使用注意事项:

- 1) 所有的语句除 SETS, ENDSETS, DATA, ENDDATA, 和 END之外必须以一个分号";"结尾.
- 2) 与 LINDO 不同的是变量可以放在约束条件的右端(同时数字也可放在约束条件的左端).
- 3) 记住用一个分号";"将目标函数与约束条件分开.
- 4) LINGO可用于求解整数规划问题. 注意在MODEL中定义一个0-1型变量用 @BIN算子, 定义一个非负整数用 @GIN 算子.
- 5) LINGO 解非线性规划时已假定各变量非负.