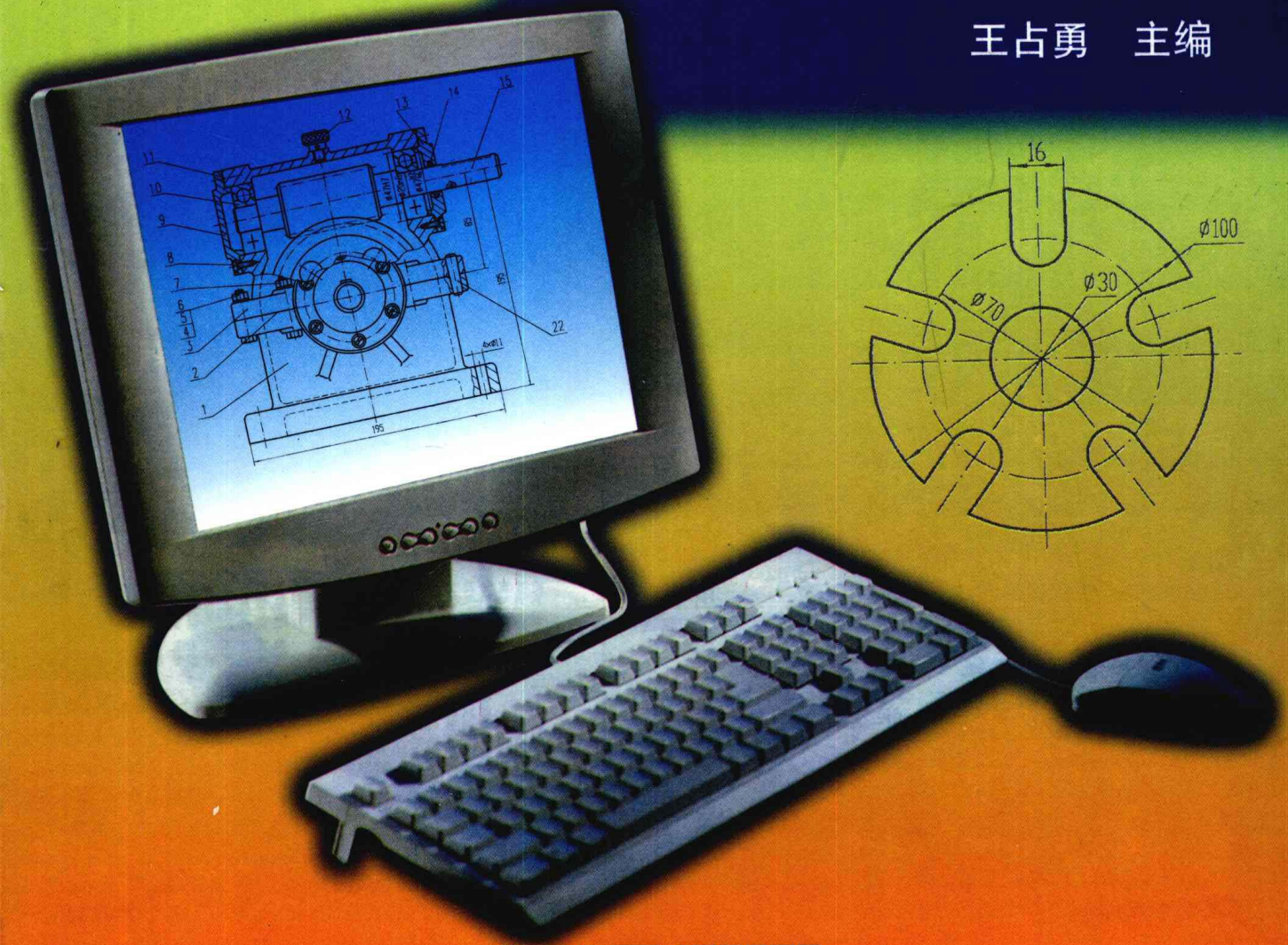
C语言典型零件CAD

王占勇　主编



NEUPRESS
东北大学出版社

C语言典型零件CAD

王占勇　主编

东北大学出版社

图书在版编目（CIP）数据

C语言典型零件CAD/王占勇主编. —沈阳：东北大学出版社，2000.9
ISBN 7-81054-554-X

Ⅰ.C… Ⅱ.王… Ⅲ.机械元件-计算机辅助设计：机械设计 Ⅳ.TH122

中国版本图书馆CIP数据核字（2000）第43305号

内 容 简 介

C语言和计算机绘图是计算机应用类课程中两门重要的课程。作者根据多年来指导机械设计课程设计的教学实践，将课程设计的科学计算和绘图内容用C语言编程和绘图函数来完成，将上述内容综合起来编写成《C语言典型零件CAD》一书。

全书包括插值和曲线拟合基础理论、工程上常用典型零件，如齿轮、轴、轴承等零件的科学计算和轴系零部件装配图以及轴、齿轮零件工作图的绘制，使读者既学习C语言编程技巧，又学习到C语言的绘图功能。

本书可作为大专院校机械类选修课程的教材，也可作为从事计算机辅助机械设计的工程技术人员的自学参考书。

（沈阳市和平区文化路3号巷11号　　邮政编码 110006）
电话：（024）23890881　　传真：（024）23892538
网址：http://www.neupress.com　E-mail:neuph@neupress.com
铁岭市新华印刷厂印刷　　东北大学出版社发行

开本：787×1092　1/16　　字数：322千字　　印张：13.25
印数：1～2600册
2000年9月第1版　　2000年9月第1次印刷

责任编辑：王兆元　　责任校对：米　戎
封面设计：唐敏智　　责任出版：秦　力

定价：18.50元

前　言

计算机辅助设计（Computer Aided Design，简称 CAD）是计算机系统在工程设计和产品设计的各个阶段和过程中，为设计人员提供各种快速、有效的工具和手段，优化设计过程和设计结果，以达到最佳的设计效果的一种技术。CAD 作为一种新的设计手段，正极大地改善和影响着传统的设计过程，在促进产品的设计朝着现代化发展中起着重要的作用。江泽民同志曾指出，“计算机辅助设计推动了几乎一切领域的设计革命……。”这说明当代大学生学习和掌握 CAD 技术是十分必要的，这是时代的需要，也是历史赋予的一种责任。

机械 CAD 是一门综合技术。作为完整的机械 CAD 系统，不是单纯的绘图，也不是单纯的工程和分析。从广义角度讲，它包含了从产品的方案决策、结构设计、性能分析、参数选择、功能仿真、事物管理，一直到工艺设计的全过程。标准化、集成化和智能化是今后机械 CAD 的发展趋势。

本书的编写目的，是在大学本科三年级机械类专业学生的机械课程设计中，让学生自己动手编程计算和绘图，以掌握 CAD 最基础的知识，并培养自己的编程能力，为今后进一步应用 CAD（如为现场进行交互式 CAD 设计或参数化设计）打下良好基础。

本书的典型零件设计程序均用 C 语言写成，其主要特点体现在如下方面：

1. 遵循结构化原则，每一系数、每一公式、每一表格均编写成独立子函数，供主函数调用。

2. 加深对设计参数的理解，从定性分析提高到定量分析。如在 V 带传动中，对计算结果略加分析，能得出：“在合理带速范围内，V 带传动功率随带速增加而提高，根数减少”的结论。在轴的计算中，轴向力的指向可以有多种组合。在没有计算前，难以判断哪种情况对轴的强度、轴承寿命最有利，只能作一般的定性分析。通过程序设计把 4 种情况计算结果进行比较，很快得出最佳方案，从而使选择齿轮的旋向就有了依据。把这种方法推广到对减速器的 3 根轴作计算分析，就得出整个部件的受力最佳方案，使设计参数的选择建立在定量分析的基础上。

3. 加深对设计理论的认识，提高分析问题、解决问题的能力。程序框图是编程的依据，它能简明扼要地说明程序的逻辑关系和数据信息的流向。绘制合理的框图，不但有助于对设计内容的逻辑关系理解深透，而且对各参数之间的关系会更清楚，这样在判断计算结果时，才能定出是与非的数据走向，合理地调整参数使之满足要求。因此程序设计是增减逻辑思维的有效手段，提高两个能力的重要途径。

4. 拓宽知识面。由于计算机适宜于复杂的数字计算，在程序中很多参数

都以其解析的形式出现，这不但提高了计算的准确性，而且拓宽了知识面。

5. 为采用新的设计方法打下基础。程序设计与设计方法学有着密切的关系，不但能实现设计过程的自动化，而且能为新的设计方法创造良好的设计环境。

6. 通过学习和实践，可以提高学生程序化的技能，进而提高 CAD 的设计能力。

本书在第 1 章和第 2 章讲述插值计算和曲线拟合等基础理论内容。因为在机械设计过程中总要引入一系列公式和大量数据资料，如理论公式、经验公式、实验曲线、图线，还有各种标准和规范等，在传统的设计中是机械设计手册提供的，但在 CAD 设计中亦应对它进行适当处理，写入程序，存入计算机，这就是所谓程序化问题。写这部分内容就是要让学生得到程序化的训练，用以解决 CAD 设计中必然遇到的问题，从而提高学生 CAD 设计能力。

本书在第 3 章到第 8 章讲述常用的 6 种零件——带、链、齿轮、蜗杆、轴和滚动轴承——的程序设计。这些零件是学生在进行课程设计时所经常接触到的，完全满足学生作课程设计的需要。

为了训练学生的编程能力，在第 5 章齿轮传动程序设计和第 6 章蜗杆传动程序设计中，只给出全部子函数和部分主函数内容，其中未完成的内容由学生自己去完成。这样既可以提高学生对计算机的应用能力，又可以加深对机械设计课程内容、概念的理解。

本书在轴程序设计一章里用很大篇幅讲述关于轴的弯矩、扭矩图绘制和轴的结构简图绘制的程序设计，其目的是为了启发学生自己开发其他绘图程序，以提高 CAD 设计能力和设计水平。

参加本书编写的有：辽宁工程技术大学王占勇（1～8 章）；王琦（9 章）；齐秀飞（10 章）。全书由王占勇担任主编，由赵灿和冷兴聚担任主审。

在本书编写过程中，得到了辽宁工程技术大学教务处单亚飞副教授和有关同志的大力支持和帮助，也得到同行专家的热情支持，作者在此表示诚挚的感谢。

限于水平，书中难免有疏漏、不妥之处，请广大读者批评、指正。

编　者

2000 年 7 月

目　录

1 函数插值理论基础

1.1 函数插值理论

在机械设计中，很多数表、函数均要求用插值方法求函数值，以满足精度方面的要求，因此，在程序中不可避免地遇到插值计算方法的问题。

插值的基本思想是：设法构造某一简单的函数 $y=p(x)$，作为数表、函数 $f(x)$ 的近似表达式，然后计算 $p(x)$ 的值，以得到 $f(x)$ 的近似值。即根据给定函数 $f(x)$ 的数表值，希望得到反映 $f(x)$ 特征而且计算比较简单的函数 $p(x)$。使

$$f(x_i)=p(x_i) \qquad i=1,2,3,\cdots,n$$

成立。$p(x)$ 是 $f(x)$ 的插值函数，点 $x_1,x_2,x_3,\cdots,x_n$ 称为插值节点。最常用的插值函数是代数多项式。多项式的次数不超过 $n-1$ 次。

1.1.1 线性插值

线性插值即两点插值，已知两端点函数 $y_1=f(x_1)$，$y_2=f(x_2)$ 构成一个 $n-1$ 次多项式即 $p_1(x)$，使它满足 $p_1(x_1)=y_1$，$p_1(x_2)=y_2$。$y=p(x)$ 的几何意义就是这两点 (x_1,y_1)，(x_2,y_2) 的直线，如图 1-1。由解析几何知插值公式

$$p_1(x)=\frac{x-x_2}{x_1-x_2}y_1+\frac{x-x_1}{x_2-x_1}y_2 \qquad (1\text{-}1)$$

可按式(1-1)编制线性插值子函数，供主函数调用。

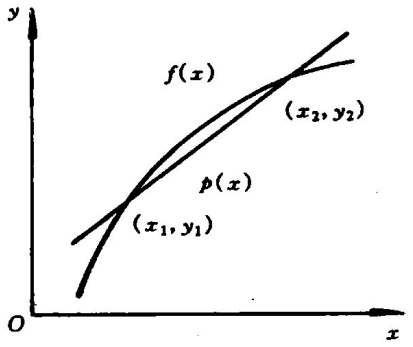


图 1-1 线性插值

1.1.2 拉格朗日插值

从式(1-1)看出，$p_1(x)$ 是两个线性函数

$$A_1(x)=\frac{x-x_2}{x_1-x_2}; \qquad A_2(x)=\frac{x-x_1}{x_2-x_1}$$

的线性组合，其系数分别是 y_1,y_2，即

$$p_1(x)=y_1A_1(x)+y_2A_2(x)$$

称 $A_1(x)$，$A_2(x)$ 为线性插值基础函数，在节点处它应满足

$$A_i(x_j)=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \qquad i,j=1,2,\cdots,n \qquad (1\text{-}2)$$

当取三个节点时，作二次多项式 $y=p_2(x)$，使它满足 $p_2(x_i)=y_i$，$i=1,2,3$。为了求出 $p_2(x)$ 的表达式，可采用基函数的方法，使其在节点处满足式(1-2)，就很容易求出基函数 $A(x)$ 的表达式。当 $x=x_1$ 时

$$A_1(x_1)=1, \quad A_2(x_1)=0, \quad A_3(x_1)=0$$

因为有两个零点，而且是二次多项式，则

$$A_1(x_1)=L(x-x_2)(x-x_3)$$

所以 $$L = \frac{1}{(x_1 - x_2)(x_1 - x_3)}$$

则 $$A_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}$$

同理可得 $$A_2(x) = \frac{(x - x_3)(x - x_1)}{(x_2 - x_3)(x_2 - x_1)}$$

$$A_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

利用二次基函数可求出二次插值多项式

$$p_2(x) = y_1A_1(x) + y_2A_2(x) + y_3A_3(x) \tag{1-3}$$

取 n 个节点，按式(1-1)及式(1-3)用基函数的方法，可求出 $n-1$ 次插值多项式

$$p_{n-1}(x) = y_1A_1(x) + y_2A_2(x) + \cdots + y_{n-1}A_{n-1}(x) + y_nA_n(x) = \sum_{j=1}^{n} y_i \left(\prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} \right) \tag{1-4}$$

式(1-4)称为拉格朗日插值多项式。当 $n=3$ 时为式(1-3)，称为抛物线插值多项式，是常用的。其程序框图如图 1-3 所示。

1.1.3　一元三点插值

一般来说，适当地提高插值公式的阶数，可以改善插值的精度。但有时出现异常。在实际插值时，常采用分段插值法。即将插值范围划分若干段，在每个分段上采用低价插值。最常用的是抛物线插值。

由式(1-4)知，当取 $k, k+1, k+2$ 三点时

$$p_{2k}(x) = \sum_{j=k}^{k+2} y_i \left(\prod_{\substack{j=k \\ i \neq j}}^{k+2} \frac{x - x_i}{x_j - x_i} \right) \tag{1-5}$$

为提高插值精度，应使所取三个节点最靠近插值点，为此 k 值应按下法来取：

$$k = \begin{cases} 0 & \text{当 } x < x_1 (s = 0, 1, 2, \cdots, n-1) \\ s & \text{当 } x_s < x < x_{s-1}, x - x_s > x_{s-1} - x \ (s = 0, 1, 2, \cdots, n-3) \\ s-1 & \text{当 } x_s < x < x_{s+1}, x - x_s > x_{s+1} - x \\ s-3 & \text{当 } x \geqslant x_{n-2} (s = 0, 1, 2, \cdots, n-1) \end{cases}$$

1.1.4　二元插值

在机械设计中也常用到二元插值，可以多次调一元插值去处理。也可以直接用二元插值公式直接计算。

(1) 二元拟线性插值可以通过二次调用一元线性插值来完成。已知函数 $Z(x, y)$ 第一变量 x 的节点 $x_i (i = 1, 2, \cdots, n)$，第二变量 y 的节点 $y_j (i = 1, 2, \cdots, m)$，可以把 y 看成不变值，用一元插值求 $Z(x, y_j)$，$Z(x, y_{j+1})$ 的函数值，然后再一次调用一元插值求 $Z(x, y)$。其几何示意图见图 1-2。

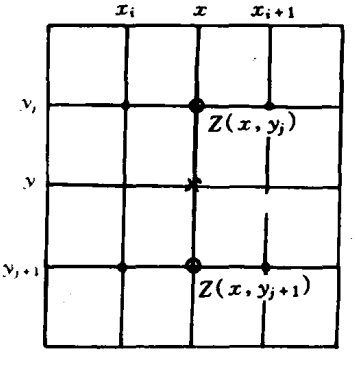


图 1-2　二元拟线几何示意图

由式(1-1)可知

$$Z(x,y_j)=Z(x_i,y_j)A_i(x)+Z(x_{i+1},y_j)A_{i+1}(x)$$

$$Z(x,y_{j+1})=Z(x,y_{j+1})A_i(x)+Z(x_{i+1},y_{j+1})A_{i+1}(x)$$

$$\begin{aligned}Z(x,y)&=Z(x,y_j)B_j(y)+Z(x,y_{j+1})B_{j+1}(y)\\&=Z(x_i,y_j)A_i(x)B_j(y)+Z(x_{i+1},y_j)A_{i+1}(x)B_j(y)+\\&\quad Z(x,y_{j+1})A_i(x)B_{j+1}(y)+Z(x_{i+1},y_{j+1})A_{i+1}(x)B_{j+1}(y)\end{aligned}\tag{1-6}$$

$$A_i(x)=\frac{x-x_{i+1}}{x_i-x_{i+1}}\qquad A_{i+1}(x)=\frac{x-x_i}{x_{i+1}-x_i}$$

$$B_j(y)=\frac{y-y_{j+1}}{y_j-y_{j+1}}\qquad B_{j+1}(y)=\frac{y-y_j}{y_{j+1}-y_j}$$

$$Z(x,y)=\sum_{Q=I}^{I+1}\sum_{P=J}^{J+1}\left(\prod_{\substack{U=I\\U\neq Q}}^{I+1}\frac{x-x_0}{x_Q-x_U}\right)\left(\prod_{\substack{U=J\\U\neq P}}^{J+1}\frac{y-y_0}{y_P-y_U}\right)Z(x_Q,y_P)$$

(2) 二元三点插值。按(1-5)形式推出二元三点插值公式

$$L(x,y)=\sum_{G=I}^{I+2}\sum_{P=J}^{J+2}\left(\prod_{\substack{U=I\\U\neq Q}}^{I+2}\frac{x-x_U}{x_Q-x_U}\right)\left(\prod_{\substack{V=I\\V\neq P}}^{J+2}\frac{y-y_V}{y_P-y_V}\right)L(x_Q,y_P)\tag{1-7}$$

1.2 程序变量名说明

表 1-1 程序变量名

序 号	变量名	原 名	含 义	单 位
1	x[ ]	x_1	插值自变量	
2	y[ ]	$f(x)$	被插函数	
3	u[ ]		插值节点	
4	v[ ]		插值函数	
5	N		y_i 值的个数	
6	M		插值节点的个数	

1.3 程序子函数一览表

表 1-2 程序子函数

序 号	子函数名	函数及形参类型	功 能
1	inserty	void inserty(x, y, u) float x[N], y[N], u[m];	插值计算
2	print	void print (x, y, u, v)	打印插值结果

1.4 程序框图

拉格朗日插值程序框图见图 1-3。

一元三点插值程序框图见图 1-4。

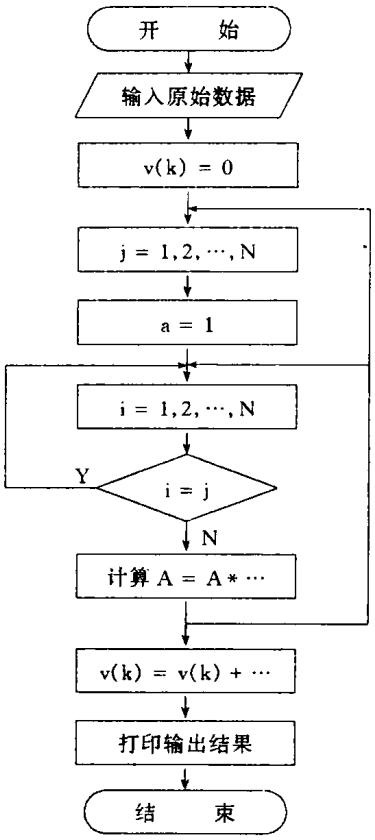


图 1-3　拉格朗日插值程序框图

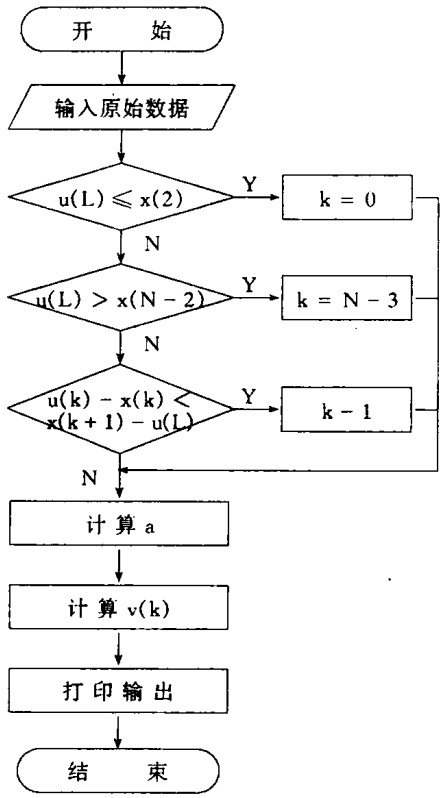


图 1-4　一元三点插值程序框图

1.5　程序使用说明

算例：

x	400	500	600	700	800	900	1000	1100	1200	1300	1400
y	0.58	0.50	0.44	0.37	0.33	0.28	0.23	0.25	0.21	0.20	0.19
u	550	950	1350								

根据算例的数据，应在文件开头宏名中定义 N = 11，M = 3，然后将 x，y 数据填入主函数 x[ ]，y[ ]，u[ ]中，就可形成下述数据文件。

```
x[0] = 400.000000        y[0] = 0.580000
x[1] = 500.000000        y[1] = 0.500000
x[2] = 600.000000        y[2] = 0.440000
x[3] = 700.000000        y[3] = 0.370000
x[4] = 800.000000        y[4] = 0.330000
x[5] = 900.000000        y[5] = 0.280000
x[6] = 1000.000000       y[6] = 0.250000
x[7] = 1100.000000       y[7] = 0.230000
x[8] = 1200.000000       y[8] = 0.210000
x[9] = 1300.000000       y[9] = 0.200000
x[10] = 1400.000000      y[10] = 0.190000
```

u[0]=550.000000　　v[0]=0.467500
u[1]=950.000000　　v[1]=0.262500
u[2]=1350.000000　　v[2]=0.1950000

1.6 源程序

```
#include "math.h"
#include "stdio.h"
#define N 11
#define M 3
#define PR printf
#define PR1 fprintf
#define RUN 0

/* 插值子函数 */
float v[M];
void inserty(x,y,u)
float x[N],y[N],u[M];
{float a;
int i,j,k,l,m,n;
n=N-1;
#ifdef RUN
for(i=0;i<=n;i++)
{PR("%f\n",x[i]);
PR("%f\n",y[i]);}
for(j=0;j<M;j++)
PR("%f\n",u[j]);
#endif
for(l=0;l<M;l++)
{
for(k=1;k<=n-2;k++)
{
m=k;
if(u[l]-x[k]<=x[k+1]-u[l]) break;}
m=m-1;
if(u[l]>=x[n-1]) m=n-2;
PR("m=%d\n",m);
for(j=m;j<=m+2;j++)
{a=1.;
for(i=m;i<=m+2;i++) {if(i==j) continue;
```

```
else
a=a*(u[l]-x[i])/(x[j]-x[i]);}
v[l]=v[l]+a*y[j];}}
for(i=0;i<M;i++)
PR("v[%d]=%f\n",i,v[i]);
}

/* 输出插值结果 */
void print(x,y,u,v)
float x[],y[],u[],v[];
{int i;
char fname[10];
FILE *fp;
PR("input filename\n");
scanf("%s",fname);
fp=fopen(fname,"w");
for(i=0;i<N;i++)
PR1(fp, "x[%d]=%f          y[%d]=%f\n",i,x[i],i,y[i]);
PR1(fp,"\n");
for(i=0;i<M;i++)
PR1(fp, "u[%d]=%f          v[%d]=%f\n",i,u[i],i,v[i]);
}
main()
{float x[N]={400,500,600,700,800,900,1000,1100,1200,1300,1400};
float y[N]={.58,.50,.44,.37,.33,.28,.25,.23,.21,.20,.19};
float u[M]={550,950,1350};
inserty(x,y,u);
print(x,y,u,v);
}
```

2 数表公式化(曲线拟合)

2.1 曲线的拟合

插值的实质是根据数表建立公式的一种方法。这种方法存在着两个明显的缺点:一是插值公式在几何上是用严格通过各个节点的曲线,来近似代替数表函数曲线。但通过实验所得到的数表函数的数据(x_i, y_i)是有离散性的,个别点离散很大,误差较大。因而用插值公式,必然保留了这些误差。二是当要求精度高时,用插值方法所得到的公式形式比较复杂,使用不方便。因此,在工程上常采用拟合的方法来构造近似曲线。此曲线并不严格通过所有节点,而是尽可能反映所给数据的趋势。这种方法称为数据的曲线拟合,或称数表公式化。曲线拟合所得到的公式称为回归方程,其理论基础为最小二乘法。

2.1.1 最小二乘法

一组数据序列(x_i, y_i), $i=1,2,\cdots,m$,用一个多项式来拟合,如图 2-1。

$$p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = \sum_{j=0}^{n} a_jx^j \tag{2-1}$$

如把 x_i 处的偏差记为

$$D_i = p_n(x_i) - y_i$$

则拟合要求其偏差 D_i 的总和最小。由于 D_i 有正负之分,因此为了真正达到最好拟合,要求各节点的偏差平方和为最小。这就是拟合问题的最小二乘法。

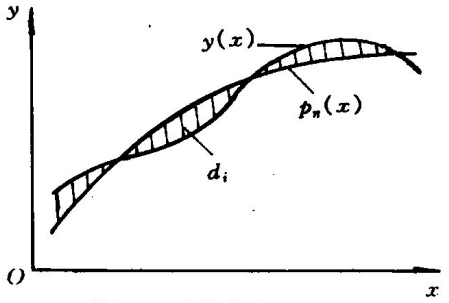


图 2-1 曲线拟合

设偏差的平方和为 $\varphi(a_0, a_1, \cdots, a_n)$

$$\varphi = \sum_{i=1}^{m} D_i^2 = \sum_{i=1}^{m} [p_n(x_i) - y_i]^2 \tag{2-2}$$

只要求出 $\varphi_{\min}$时的 $a_j(j=0,1,\cdots,m)$,代入式(2-1),所得到的 $p_n(x)$即为偏差平方和极小的拟合方程。由此可见,曲线拟合可归结为多元函数求值问题。即

$$\frac{\partial \varphi}{\partial a_k} = 0 \qquad (k = 0, 1, 2, \cdots, n)$$

$$\frac{\partial \varphi}{\partial a_k} = \sum_{i=1}^{m} 2\Big(\sum_{j=0}^{n} a_jx_i^j - y_i\Big)\frac{\partial \varphi}{\partial a_k}\sum_{j=0}^{n} a_jx_i^j$$

$$= 2\sum_{i=1}^{m}\Big(\sum_{j=0}^{n} a_jx_i^j - y_i\Big)x_i^k = 2\Big(\sum_{j=0}^{n} a_j\sum_{i=1}^{m} x_i^{j+k}\Big) - \sum_{i=1}^{m} y_jx_i^k$$

令

$$S_L = \sum_{i=1}^{m} x_i^L \qquad (L = 0, 1, 2, \cdots, 2n)$$

$$t_k = \sum_{i=1}^{m} x_i^k \qquad (k = 0, 1, 2, \cdots, n)$$

则

$$\frac{\partial \varphi}{\partial a_k} = 2\left(\sum_{j=0}^{n} a_j s_{j+k} - t_k \right) = 0$$

$$\sum_{j=0}^{n} a_j s_{j+k} = t_k \qquad (k = 0, 1, 2, \cdots, n)$$

写成线性方程形式

$$\left.\begin{aligned} & s_0 a_0 + s_1 a_1 + \cdots + s_n a_n = t_0 \\ & s_1 a_0 + s_2 a_1 + \cdots + s_{n+1} a_n = t_1 \\ & \cdots\cdots\cdots\cdots\cdots\cdots \\ & s_n a_0 + s_{n+1} a_1 + \cdots + s_{2n} a_n = t_n \end{aligned}\right\} \tag{2-3}$$

解方程组(2-3)求出 $a_j(j=0,1,2,\cdots,n)$，代入式(2-1)即可求出 $p_n(x)$的表达式。在实际应用时一般取 $n<m$。当 $n=m$ 时，求得多项式为拉格朗日插值多项式。当 $n>m$ 时，没有确定解。

2.1.2 回归方程的常用类型

回归方程常用的类型有线性回归、抛物线回归、可化为线性回归的曲线回归和多元回归等。下面介绍常用的几种。

(1) 线性回归及抛物线回归。

$$y = a_0 + a_1 x$$

$$y = a_0 + a_1 x + a_2 x^2$$

显然这两个方程可用上述方法直接求解。

(2) 可化为线性回归的曲线回归。在工程实验中的数据序列是多样的，把这些数据画在坐标纸上，呈现各种形状，如疲劳曲线、滑动轴承中与轴承特性曲线 S 有关的曲线。如用多项式去构造近似曲线，难以满足精度要求，这时根据数据的趋势，采用适当的变量替换，常使问题得以解决。

① 幂回归。

$$y = ax^b \qquad (b \text{ 为实数}) \tag{2-4}$$

$$\ln y = \ln a + b \ln x$$

$$p_1(x) = a_0 + a_1 x$$

式中　$p_1(x) = \ln y, a_0 = \ln a, a_1 = b, x = \ln x$。

拟合求出 a_0, a_1，则

$$a = e^{a_0}, \qquad b = a_1$$

得到了拟合方程

$$y = ax^b$$

② 指数回归。

$$y = ae^{bx} \tag{2-5}$$

$$\ln y = \ln a + bx$$

$$p_1(x) = a_0 + a_1 x$$

式中 $p_1(x)=\ln y$, $a_0=\ln a$, $a_1=b$, $x=x$。

同理可求拟合方程。

2.2 程序变量名说明

表 2-1 程序变量名

序 号	变量名	原 名	含 义	单 位
1	x[m]	x_i	表列 x 值	
2	y[m]	y_i	表列 y 值	
3	m[ ]		$x(y)$值的个数	
4	g[ ]		控制变量	

2.3 程序子函数一览表

表 2-2 程序子函数

序 号	子函数名	函数及形参类型	功 能
1	hab	float hab(a, c, i, j, n) float a, c; int i, i, n;	两行相减
2	bc	float bc(a, c, j, n) float a, c; int j, n;	一行乘(1/a[j][i])
3	abv	float abv(a, j, n)	换行
4	solu	float solu(a, n)	解方程组
5	print	void print(变量类型见 2.2)	打印输出结果

2.4 程序框图

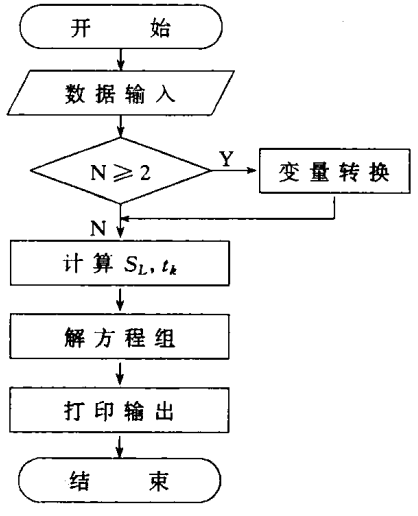


图 2-2 曲线拟合程序框图

2.5 程序使用说明

(1) 程序按式(2-1),(2-2),(2-3),(2-4),(2-5)编写子函数和主函数。

(2) 程序采用一控制变量 g。g 的取值如下:

$$g=\begin{cases}0, & y_0=ae^{bx}\\ 1, & y_1=ax^b\\ 2, & y_2=a_0+a_1x\\ 3, & y_3=a_0+a_1x+a_2x^2\end{cases}$$

使用程序时,只要输入相应 g 的值,程序就会输出拟合方程表达式及其相应的拟合值。

(3) 如果取拟合方程的阶数增加,则程序中应参考宏名 S,U 的值。

2.6 源程序

```
#include "math.h"
#include "stdio.h"
#define D "%d"
#define F "%10.2f%10.2f%10.5f\n"
#define PR printf
#define FPR fprintf
#define N "\n"
/* #define M 11 */
#define NN 0
#ifdef NN
#define S 4
#define U 3
#else
#define S 3
#define U 2
#endif

/* 消元(两行相减)子函数 */
float hab(a,c,i,j,n)
int n,i,j;
float a[][S],c;
{int s,l;
l=n+1;
for(s=j;s<=l;s++) a[i][s]=a[i][s]+c*a[j][s];
return a[i][s];
}
```