《计算机数控系统》

课程实验指导书

电子信息与电气技术实验中心

2015 年 2 月

一、概 述

机床数控的核心问题就是控制刀具和工件的运动以形成预定的运动轨迹。在数控加工中，对于复杂型面，一般采用小段直线或圆弧（即基本线型），在满足精度要求的条件下，逐段拟合高次曲线的轮廓，这种拟合的方法，就是插补。

能完成插补工作的装置叫插补器，由数字电路完成插补工作的叫做硬件插补，由软件实现插补工作的叫做软件插补。

插补的方法很多，主要分为脉冲增量插补法和数字增量插补法两类。对于脉冲增量插补，数控装置向有关运动坐标输出进给指令脉冲，每个脉冲代表最小位移，指令脉冲序列的频率代表坐标运动速度，脉冲的数量代表位移量；对于数字增量插补，数控装置向有关坐标输出进给指令数据（位置增量值），插补分粗插补和精插补两步完成，粗插补时用微小直线段来逼近给定曲线，精插补时对微小线段进行脉冲增量插补。

逐点比较法和数字积分法属于脉冲增量插补法，是掌握数控插补原理的基础，适用于精度和速度要求不高的数控系统中，本实验旨在测试和调试脉冲增量插补法。

二、实验设备

1．装有 Windows XP 的操作系统的微机　　1 台

2．C-Free 软件和 Easy Graphics Engine 函数库　　1 套

3．嵌入式系统（备用）　　1 套

三、实验准备知识

1．PC 微机系统

在 WindowsXP 操作系统的微机上，安装 C-Free 软件和 Easy Graphics Engine 函数库。C-Free 的 IDE 开发环境，采用 C 语言，设计并编写数控插补算法，结合 EGE 图形函数库进行绘制插补路径，从而实现验证插补算法。

2．实验原理

在实际加工过程中，插补的执行机构由电机来实现。本实验旨在测试与研制插补算法，因此采用图形轨迹来模拟电机执行机构的运动轨迹，从而实现插补算法轨迹控制的验证。

2.1 计算机窗口坐标系

计算机屏幕坐标系的坐标原点在屏幕的左上角，向右为 X 方向，向下位 Y 方向，坐标系中的单位是屏幕的最小分辨率率单位，即像素点，取值范围只能为整数，具体的取值范围与频幕的分辨率有关。对于分辨率设置为 1024*768 的显示器而言，X 轴的坐标取值为[0-1023]，Y 轴的取值范围为[0-767]。

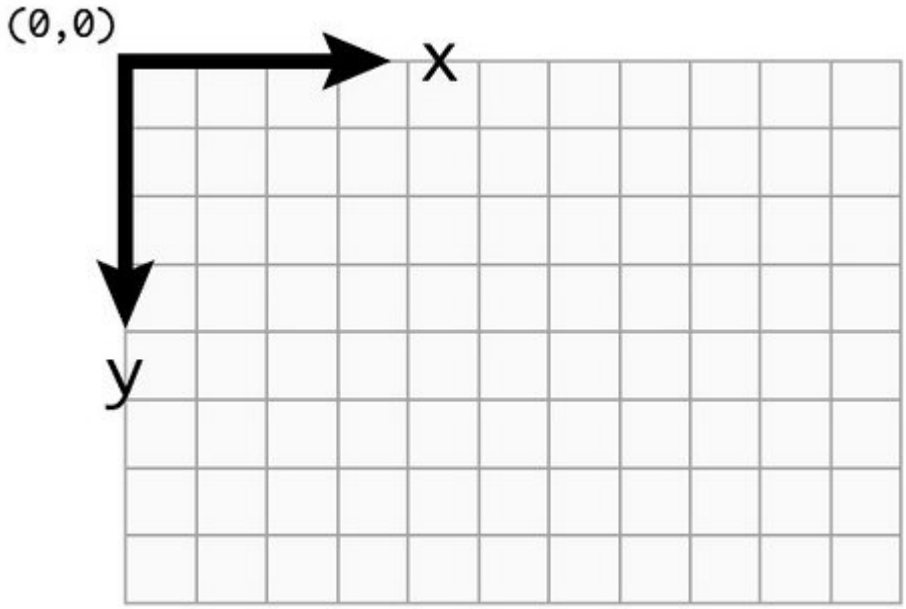

图 1　计算机屏幕坐标系

2.2 Easy Graphics Engine 函数库常用基本函数

（1）图形初始化函数

void EGEAPI initgraph(int Width, int Height, int Flag = INIT_DEFAULT);　　// 初始化图形环境

调用此函数可以初始化图形环境，例如初始化一个 800*800 的窗口，initgraph(800,800)。

（2）关闭图形环境

void EGEAPI closegraph();　　// 关闭图形环境

程序结束的时候，关闭图形环境，释放资源。

（3）设置当前绘图背景色

void EGEAPI setbkcolor(COLORREF color, PIMAGE pimg = NULL);　　// 设置当前绘图背景色（设置并做背景色像素替换）

（4）设置当前绘图前景色

void EGEAPI setcolor(COLORREF color, PIMAGE pimg = NULL);　　// 设置当前绘图前景色

（5）绝对坐标画线

void EGEAPI line(int x1, int y1, int x2, int y2, PIMAGE pimg = NULL);　　// 画线，起点（x1,y1）,终点（x2,y2）

(6)画圆弧

void EGEAPI arc(int x, int y, int stangle, int endangle, int radius, PIMAGE pimg = NULL);　// 画圆弧

x,y 为所绘制的弧线以(x,y)为圆心。

radius:所绘制弧线的半径，从 stangle 开始到 endangle 结束（用度表示），画一段圆弧线。在 EGE 中规定 x 轴正向为 0 度，逆时针方向旋转一周，依次为 90 度、180 度、270 度和 360 度。

（7）指定位置输出文字

void EGEAPI outtextxy(int x, int y, LPCSTR　textstring, PIMAGE pimg = NULL);　// 在指定位置输出文字

（8）绘制圆形

void EGEAPI circle(int x, int y, int radius, PIMAGE pimg = NULL);

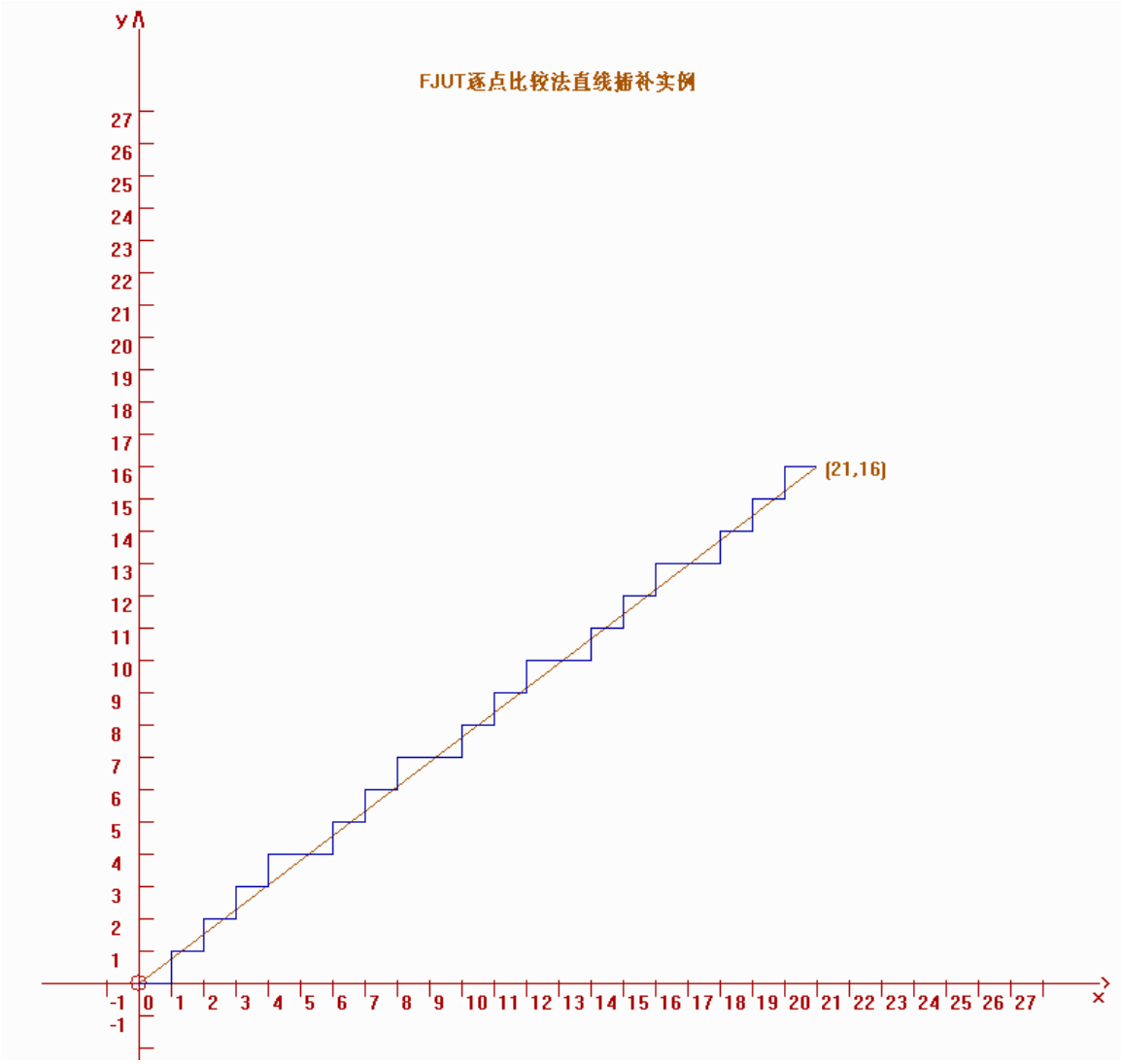

图 2　逐点比较法直线插补轨迹图

上机练习　逐点比较法直线插补控制程序设计与调试练习

1．练习目的

(1) 掌握逐点比较法的直线插补原理；

(2) 熟悉 C/C++语言；

(3) 根据样例程序，调试插补控制程序。

2．练习内容

采用 C 语言，参考给出的逐点比较法直线插补程序，控制计算机在给定的坐标系中，以坐标单位绘出直线插补的轨迹，通过控制延时来控制轨迹绘制速度。

3．步骤

(1) 熟悉在 PC 机上 C/C++编程操作；

(2) 观察逐点比较法直线插补算法的程序实现结构；

(3) 熟悉并掌握逐点比较法的算法实现过程；

(4) 调试逐点比较法直线插补程序，了解坐标的绘制过程和轨迹的产生过程；

(5) 操作练习，修改插补终点值（不超过（27,27)），绘出直线插补的轨迹图，观察是否与理论路径一样，通过实验，计算逐点比较法的最小插补精度是多少？

实验一　逐点比较法圆弧插补控制程序设计与调试实验

1．实验目的

(1) 掌握逐点比较法的圆弧插补原理；

(2) 熟悉 C/C++语言；

(3) 采用 C/C++语言，设计、编制并调试圆弧插补控制程序。

2．实验内容

采用 C/C++语言，设计逐点比较法圆弧插补程序，控制计算机在坐标系中绘制逐点比较法圆弧插补路径，绘制出圆弧插补的轨迹，并观察插补实现过程。

3．实验步骤

(1) 掌握逐点比较法圆弧插补原理；

(2) 使用 C/C++语言进行程序设计；

(3) 参考逐点比较法直线插补算法的程序实现结构，按圆弧插补算法的实现原理，修改程序；

(4) 调试逐点比较法圆弧插补程序，观察圆弧插补轨迹的产生过程；

4．实验报告

(1) 在实验报告上写明实验目的、内容、原理和步骤；

(2) 绘出与所设计程序相一致的逐点比较法圆弧插补程序的框图；

(3) 电脑打印出所调试的圆弧插补程序的清单，并注释各语句的含义；

(4) 附出圆弧插补程序的轨迹图；

(5) 总结实验内容，说明实验收获。

实验二　数字积分法直线插补控制程序设计与调试实验

1．实验目的

(1) 掌握数字积分法的插补原理；

(2) 熟悉 C/C++语言；

(3) 采用 C/C++语言，设计、编制并调试 DDA 直线插补控制程序。

2．实验内容

采用 C/C++语言，设计数字积分法直线插补程序，控制计算机在坐标系中绘制数字积分法直线插补路径，绘制出直线插补的轨迹，并观察插补实现过程。

3．实验步骤

(1) 掌握数字积分法直线插补原理；

(2) 使用 C/C++语言进行程序设计；

(3) 参考数字积分法直线插补算法的程序实现结构，按直线插补算法的实现原理，修改程序；

(4) 调试数字积分法直线插补程序，观察直线插补轨迹的产生过程；

4．实验报告

(1) 在实验报告上写明实验目的、内容、原理和步骤；

(2) 绘出与所设计程序相一致的数字积分法直线插补程序的框图；

(3) 电脑打印出所调试的程序清单，并注释各语句的含义；

(4) 附出直线插补程序的轨迹图；

(5) 总结实验内容，说明实验收获。

附 录

附录 1： 逐点比较法直线插补应用程序——供参考

```
#include<stdio.h>
#include <graphics.h>
#include<conio.h>
#include<time.h>
#include<math.h>
#include<stdlib.h>
#define L 23                          //标度宽度
#define NUM 30                        //标度个数
const int X=100;
const int Y=700;                      //原点坐标

int Xe;   //X 终点坐标
int Ye; //Y 终点坐标（终点坐标小于 NUM-3）
int counter_length;     //n 步数计数
int deviation_function;     //Fi 偏差判别

int Xi=0; //动点坐标 X
int Yi=0; //动点坐标 Y

void xy()
{
    int i;
    char *nr[]={"-1","0","1","2","3","4","5","6","7","8","9","10","11",

"12","13","14","15","16","17","18","19","20","21","22",
                "23","24","25","26","27","28","29","30","31","32","33",
                "34","35","36","37","38","39","40","41","42","43","44"};
    setcolor(RED);//设置红色圆点
    circle(X,Y,6);//基于原点绘制一个半径 6 的圆形
    setcolor(RED);
    line(X-L*3,Y,X+L*NUM,Y);// 绘 制 一 条 （ X-L*3 ， Y ） ----> （X+L*NUM,Y）的直线
    outtextxy(X+L*NUM-5,Y-8,">");//在(X+L*NUM-5,Y-8)的位置上输出一个箭头>
    outtextxy(X+L*NUM-10,Y+1,"x");//在(X+L*NUM-10,Y+1)的位置上输出一个 X
    line(X,Y+L*3,X,Y-L*NUM);// 绘 制 一 条 (X,Y+L*3 ） ----> （X,Y-L*NUM)的直线
    outtextxy(X-16,Y-L*NUM-5,"y /\\");
    //Sleep(1000);
    for(i=0;i<NUM;i++)
    {
```

```
        line(X-L+i*L,Y-2,X-L+i*L,Y+10);//打 X 轴的小刻度
        line(X+10,Y+2*L-i*L,X,Y+2*L-i*L);//打 Y 轴的小刻度
    }
    for(i=0;i<29;i++)
    {
        outtextxy(X-L+i*L+3,Y+6,nr[i]);//标 X 轴刻度值
        if(i!=1)
        outtextxy(X-20,Y+L-i*L-1,nr[i]);//标 Y 轴刻度值，取消 0 重复
标

    }

}

//X 轴走一步， 根据给点坐标，绘制 X 轴路径
//X 轴的路径从上一点开始绘制，因此要记录上一点的 X 轴坐标和 Y
轴坐标
void x_feed_1_step(void)
{ int Xt,Yt;//记录动点坐标变化中间值
   Xt=Xi+1; Yt=Yi;
   setcolor(BLUE);
   line(X+Xi*L,Y-Yi*L,X+Xt*L,Y-Yt*L);//绘制当前点到新动点的直线

   Xi=Xt;Yi=Yt;
   deviation_function -=Ye;
}

//Y 轴走一步， 根据给点坐标，绘制 Y 轴路径
//Y 轴的路径从上一点开始绘制，因此要记录上一点的 X 轴坐标和 Y
轴坐标
void y_feed_1_step(void)
{ int Xt,Yt;//记录动点坐标变化中间值
   Xt=Xi; Yt=Yi+1;
   setcolor(BLUE);
   line(X+Xi*L,Y-Yi*L,X+Xt*L,Y-Yt*L);//绘制当前点到新动点的直线
   Xi=Xt;Yi=Yt;
   deviation_function +=Xe;
}

                                        //制作 x-y 坐标系
int    main()
{

    char XYe[100];
     initgraph(800, 800);
    setbkcolor(WHITE);

    setcolor(BROWN);
    outtextxy(300, 50, "FJUT 逐点比较法直线插补实例");
```

```
    xy();   //绘制坐标系

   Xe =21;   //设定终点坐标
   Ye =16;
   sprintf(XYe,"(%d,%d)",Xe,Ye);
   counter_length =Xe+Ye;    //JN =JX+JY    步数计数
   deviation_function =0;

setcolor(BROWN);//先绘制出起点和终点的规划路径直线
line(X,Y,X+Xe*L,Y-Ye*L);
printf("X=%d,Y=%d,X+Xe*L=%d,Y-Ye*L=%d\n",X,Y,X+Xe*L,Y-Ye*L
);
outtextxy(X+Xe*L+6,Y-Ye*L-6,XYe);
//逐点比较法直线插补
loop1: if (deviation_function < 0)
   {
    y_feed_1_step();
    Sleep(500);
   }
           else
   {
   x_feed_1_step();
    Sleep(500);
   }
   counter_length --;
   if (counter_length >0) goto loop1;

        getch();
    closegraph();
    return 0;
}
```