國立中正大學機械工程學系

數值方法

Lab6

機械四 408420030 李晉碩

中 華 民 國 １１１ 年 １２ 月 ３０ 日

目錄

一、解題過程

1.1 迭代路徑計算

這次的題目跟工具機的控制模擬有關，是要我們模擬輪廓控制系統，給定一個固定路徑，使用不同的 feed rate，輸出經過控制系統後的路徑，並觀察與原始路徑的誤差。

首先要先定出每個轉角的點座標，第一個點取 x=10 和 y=10 為起點，逆時針依序定出座標，z 代表是否為走圓形路徑的記號，0 代表走直線，1 則為走圓形，依序將 10 個點定出來。

```
int main(int argc, char* argv[])
{
    cPt[0].x = 0;
    cPt[0].y = 10;
    cPt[0].z = 0;

    cPt[1].x = 20;
    cPt[1].y = 10;
    cPt[1].z = 1;
```

定完座標後，輸入 feed rate 決定進給的速度。

```
double v;
cout << "輸入V =\n";     //------------輸入feedrate
cin >> v;
```

接著使用路徑差補的迭代公式計算出點座標，分別使用直線和半圓曲線公式分開計算微分項 dCdt。

```
for (i = 0; i < 10; i++) {
    Point pt1, pt2, ptt, ptt2;
    double ui = 0;
    double dCdt;
    pt1 = cPt[i];
    pt2 = cPt[i + 1];

    if (cPt[i].z == 0) {
        dCdt = sqrt(pow((pt2.x - pt1.x), 2) + pow((pt2.y - pt1.y), 2));
    }
    else {
        dCdt = PI * r;
    }
```

接著從 u=0 開始使用迭代公式至 u>1 時停止。

直線公式:

$$C(u)=(1-u)*P1+u*P2$$

半圓公式:

$$C(u)=[x0 + r*\cos(u+1)\,\pi\,,\ y0 + r*\sin(u+1)\,\pi\,]$$

```
//---------------------------------實際路徑
double ui2 = 0;
while (ui2 <= 1) {
    if (cPt[i].z == 0) {
        path[t2].x = (1 - ui2) * pt1.x + ui2 * pt2.x;
        path[t2].y = (1 - ui2) * pt1.y + ui2 * pt2.y;
    }
    else {
        path[t2].x = (pt1.x + pt2.x) / 2 + r * cos((ui2 + 1) * PI);
        path[t2].y = (pt1.y + pt2.y) / 2 + r * sin((ui2 + 1) * PI);
    }
    ui2 = ptt2.inputFeedrate(v,ui2, dCdt);  //-------------------------迭帶下個Ui

    //cout << "finalPoint2=" << finalPoint2 << "  x=" << path[t2].x << "    y=" <<
    finalPoint2 = t2;
    t2++;
}
```

計算 Ui 的迭代公式，v 為可變輸入的 feed rate，迭代每次新的 Ui 值，重新計算 x 點和 y 點。

```
double Point::inputFeedrate(double v,double ui2, double dcPt) {
    double Ts = 5 * pow(10, -4);
    double uii = ui2 + v * Ts / dcPt;
    return uii;
}
```

1.2 State space

要算出 State space，我們要先化簡系統的轉移函數，這裡老師說可以用手算或其他方法，因此就用 Matlab 算出來後再帶入計算。State space 帶入公式:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} -a & -b & -c \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{B} u$$

$$y = \underbrace{\begin{bmatrix} d & e & f \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{[0]}_{D} u$$

轉移函數的計算先手動化簡係數，將各函數進行串聯、並聯和回授計算，然後計算轉移函數 ：

```
Command Window

  GX =

          0.007711 s^2 + 3.568 s + 360.2
    ----------------------------------------------
    3.62e-05 s^3 + 0.01823 s^2 + 4.716 s + 360.2

  Continuous-time transfer function.


  GY =

          0.008123 s^2 + 3.744 s + 375.9
    ----------------------------------------------
    3.817e-05 s^3 + 0.01914 s^2 + 4.944 s + 375.9

  Continuous-time transfer function.
```

```
10      %%---------X項
11 —    Kt=0.423;
12 —    J=3.62*(10^-5);
13 —    B=2.806*(10^-3);
14 —    Kpp=156.881;
15 —    Kvp=5.727*(10^-2);
16 —    Kvi=8.527;
17 —    Kf=0.5;
18 —    Ts=5*10^-4;
19 —    Kb=2/3.1415926;
20
21 —    q1=Kf*Kvp*Kt*Kb;
22 —    q2=(Kf*Kvi+Kpp*Kvp)*Kt*Kb;
23 —    q3=Kpp*Kvi*Kt*Kb;
24 —    q4=J;
25 —    q5=B+Kvp*Kt*Kb;
26 —    q6=(Kvi+Kpp*Kvp)*Kt*Kb;
27 —    q7=Kpp*Kvi*Kt*Kb;
28 —    GX=tf([q1 q2 q3],[q4 q5 q6 q7])
```

State space 計算:

```
void statespace() {
    double a0 = 3.62 * pow(10, -5);
    double a = -0.01823 / a0;
    double b = -4.716 / a0;
    double c = -360.2 / a0;
    double d = 0.007711/a0;
    double e = 3.568/a0;
    double f = 360.2/a0;
    double A[3][3] = { {a,b,c},{1,0,0},{0,1,0} };
    double B[3][1] = { 1,0,0 };
    double C[1][3] = { d,e,f };
```

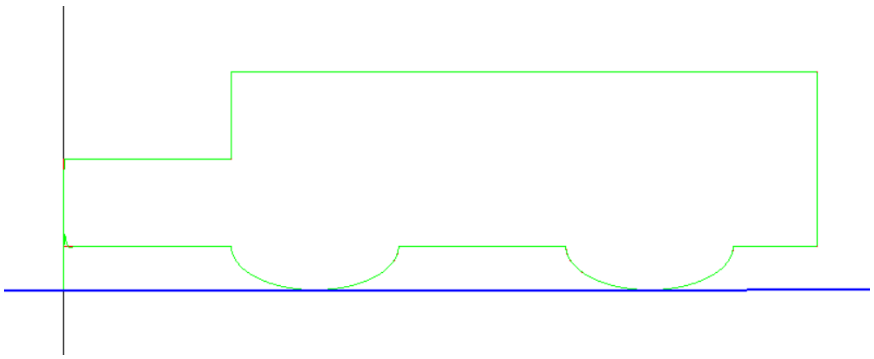
```
for (double t = 0; t < TotalTime; t = t + h) {
    double xii[3][1] = { 0 };
    double y = 0;
    for (i = 0; i < 3; i++) {
        xii[0][0] += A[0][i] * xi[i][0];
        xii[1][0] += A[1][i] * xi[i][0];
        xii[2][0] += A[2][i] * xi[i][0];
    }
    xii[0][0] = h * (xii[0][0] + u * B[0][0]) + xi[0][0];
    xii[1][0] = h * (xii[1][0]) + xi[1][0];
    xii[2][0] = h * (xii[2][0]) + xi[2][0];
    for (i = 0; i < 3; i++) {
        y += C[0][i] * xi[i][0];
    }
    //cout << y << endl;
    for (i = 0; i < 3; i++) {
        xi[i][0] = xii[i][0];
    }
    output[num] = y;
    //cout << "t=" << t << "   y=" << output << "\n";
    numFinal = num;
    num++;
    u = 1;
}
```

二、結果與討論

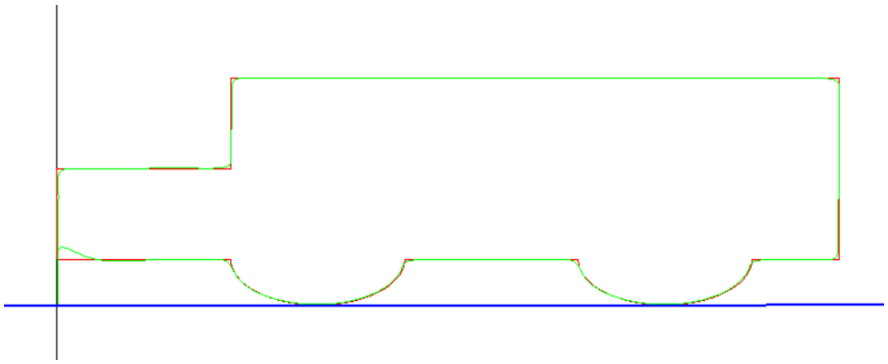
2.1 程式計算結果

測試不同的 feed rate 的路徑與原始路徑的誤差，原始路徑為紅色，實際路徑為綠色。我們可以觀察到，在轉角處誤差會比較大，直線誤差較小，當 V 越大時此情況會愈明顯。

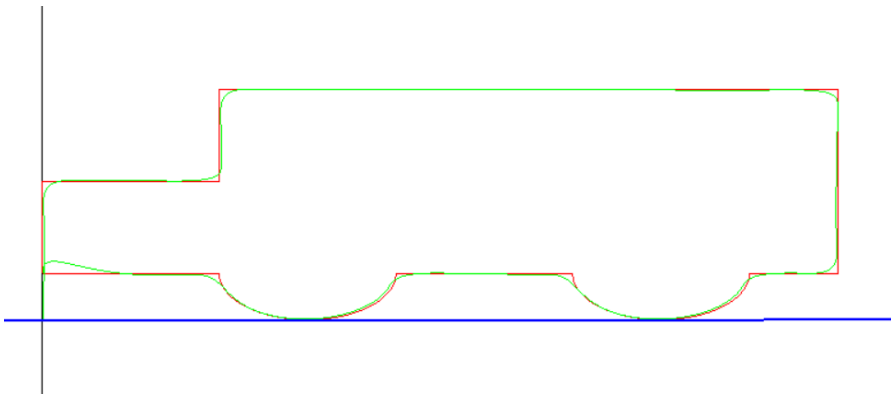
(1) V=10



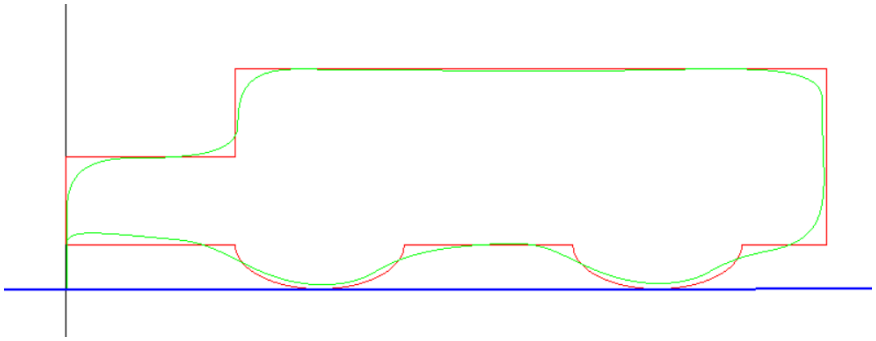
(2) V=100



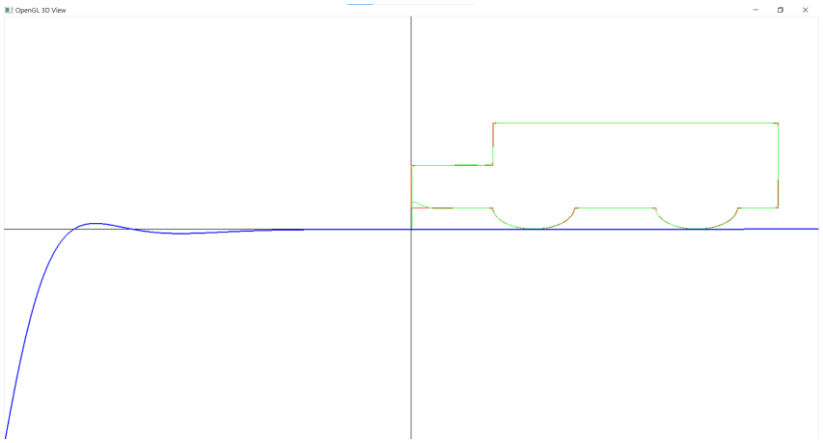
(3) V=200



(4) V=500



(5) Step response



Step response 算出來會趨近於 1，在原座標上不明顯，因此經過平移放大後結果如圖。

2.2 討論

這次做出來的圖形感覺還算成功，但是起點的部分不太確定是不是正常的，因為當時看老師的圖形也是差不多這樣，起點已經確認過是從左下角(0,10)開始，但往下個點移動會有一個往上偏的趨勢。其他部分轉彎處的路徑都是正常的，直線也都很精準幾乎沒誤差。

2.3 心得

這次程式也成功做出來，但有些地方還有改進的空間。像是系統的轉移函數，可以使用 C++程式寫出來，但是怕花太多時間，因此選擇用其他方法算，這部分可以做得更好。此外參數的部分有很多類似的項目，寫到最後自己都有點搞混，更不用說讓人看懂，本身還在研究物件導向的東西，因此這方面就沒用太多，未來還可以改進，讓程式的易讀性更佳。還有動畫的部分，自己在測試時有成功做出來一點，但是做的時間偏少，沒來得及弄完全部，實屬有點可惜。

透過這次作業，我們學習如何使用數值方法模擬控制系統，數值方法也廣泛應用於工具機，CNC 對於工具機的座標運動進行控制，在加工路徑的計算，最重要的是保證運動精度和定位精度，而我們可以利用差補法則的原理解決誤差問題。差補法則是指在工具機加工中，用來平衡主軸與副軸的運轉速度差異的方法，這種方法可以減少加工時因為速度差異產生的問題，使得加工精度和效率更高。未來我們也可以透過撰寫程式的方式，模擬刀具的加工路徑和誤差等等的計算，算是非常實用的方法。

上完這學期的課學到了很多實用的知識，也增進了自己的程式撰寫能力，透過這門課也更了解數值方法在各領域的應用，還聽到老師分享許多非常受用的經驗，感謝老師和助教這一學期的幫助。