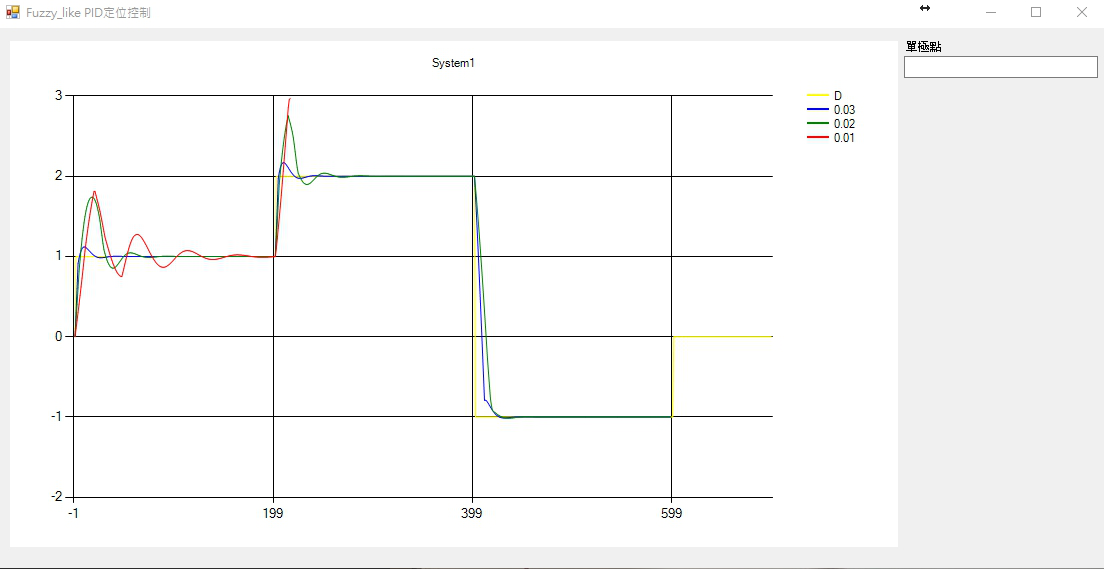
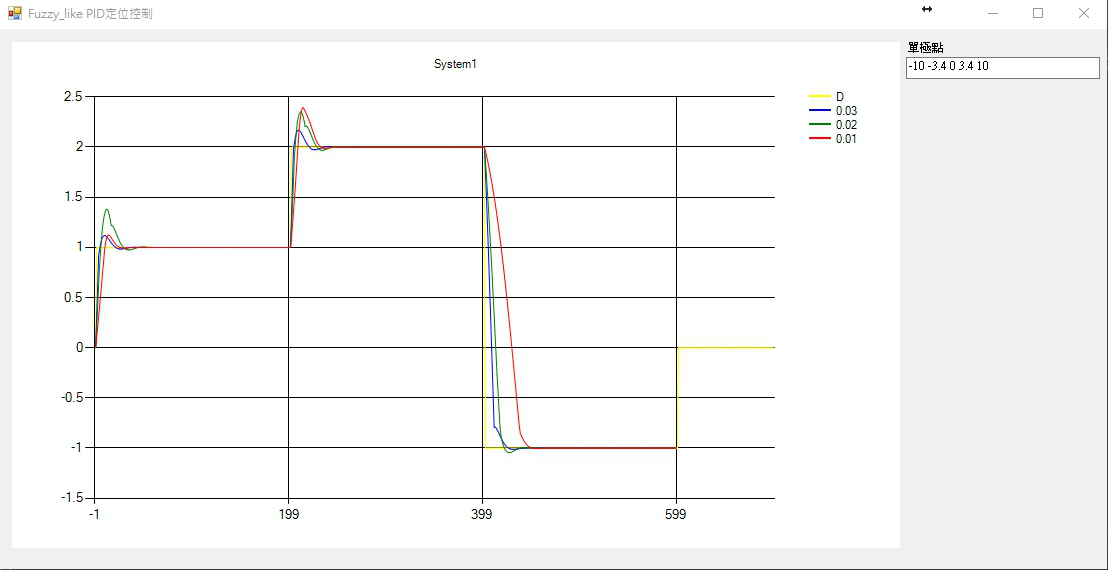
結果:



結果(加分):



程式碼

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public static Double D = 1;

public static int MAX = 700;

private void Form1\_Load(object sender, EventArgs e)

{

Text = "Fuzzy\_like PID定位控制";

System1(chart\_out);

}

public class E

{

public Double e = 0;

public Double e\_dot = 0;

public E(Double e, Double e\_dot)

{

this.e = e;

this.e\_dot = e\_dot;

}

public E()

{

this.e = 0;

this.e\_dot = 0;

}

}

public class X

{

public Double x1 = 0;

public Double x2 = 0;

public X(Double e, Double e\_dot)

{

this.x1 = e;

this.x2 = e\_dot;

}

public X()

{

this.x1 = 0;

this.x2 = 0;

}

}

public class Fuzzy

{

private static String[,] table ={

{ "NB", "NB", "NB", "NS", "PB" } ,

{ "NB", "NS", "NS", "PS", "PB" } ,

{ "NB", "NS", "ZE", "PS", "PB" } ,

{ "NB", "NS", "PS", "PS", "PB" } ,

{ "NB", "NS", "PB", "PB", "PB" } };

public static Double[] Single\_pole = { -10, -0.8, 0, 0.8, 10 };

private static Double[] Y1 = { 1, 0, 1, 0, 1 };

private static Double[] Y2 = { 0, 1, 0, 1, 0 };

private static Double[] num\_A = { -1, -0.5, 0, 0.5, 1 };

private static Double[] num\_A\_dot = { -0.3, -0.1, 0, 0.1, 0.3 };

public static Double Convert\_Function(String str)

{

switch (str)

{

case "NB":

return Single\_pole[0];

case "NS":

return Single\_pole[1];

case "ZE":

return Single\_pole[2];

case "PS":

return Single\_pole[3];

case "PB":

return Single\_pole[4];

default:

return Single\_pole[0];

}

}

public static Double get\_table\_A(int i, int j)

{

return Convert\_Function(table[i, j]);

}

public static int Rank\_D\_A(Double x)

{

Double[] distanse = new Double[5];

for (int i = 0; i < 5; i++)

{

distanse[i] = Math.Abs(x - num\_A[i]);

}

Double min = distanse[0];

int index = 0;

for (int i = 1; i < 5; i++)

{

if (distanse[i] < min)

{

min = distanse[i];

index = i;

}

}

if (index == 4) return 3;

return (index);

}

public static int Rank\_D\_A\_dot(Double x)

{

Double[] distanse = new Double[5];

for (int i = 0; i < 5; i++)

{

distanse[i] = Math.Abs(x - num\_A\_dot[i]);

}

Double min = distanse[0];

int index = 0;

for (int i = 1; i < 5; i++)

{

if (distanse[i] < min)

{

min = distanse[i];

index = i;

}

}

if (index == 4) return 3;

return (index);

}

public static E FuzzyControl\_A(int k, Double y\_next, Double e)

{

Double e\_next, e\_dot\_next;

e\_next = CalculateNextK\_A(k) - y\_next;

e\_dot\_next = e\_next - e;

return new E(e\_next, e\_dot\_next);

}

public static Double CalculateU\_A(Double e, Double e\_dot, Double u\_previos)

{

int zone\_e = -1, zone\_e\_dot = -1;

Double u1, u2, u3, u4;

zone\_e = Rank\_D\_A(e);

zone\_e\_dot = Rank\_D\_A\_dot(e\_dot);

if (Y1[zone\_e] > Y1[zone\_e + 1])

{

if (e\_dot <= num\_A[0]) u1 = 1;

else if (e\_dot >= num\_A[4]) u1 = 0;

else u1 = Math.Abs(e - num\_A[zone\_e + 1]) / Math.Abs(num\_A[zone\_e] - num\_A[zone\_e + 1]);

u2 = 1 - u1;

}

else

{

u2 = Math.Abs(e - num\_A[zone\_e]) / Math.Abs(num\_A[zone\_e] - num\_A[zone\_e + 1]);

u1 = 1 - u2;

}

if (Y1[zone\_e\_dot] > Y1[zone\_e\_dot + 1])

{

if (e\_dot <= num\_A\_dot[0]) u3 = 1;

else if (e\_dot >= num\_A\_dot[4]) u3 = 0;

else u3 = Math.Abs(e\_dot - num\_A\_dot[zone\_e\_dot + 1]) / Math.Abs(num\_A\_dot[zone\_e\_dot] - num\_A\_dot[zone\_e\_dot + 1]);

u4 = 1 - u3;

}

else

{

u4 = Math.Abs(e\_dot - num\_A\_dot[zone\_e\_dot]) / Math.Abs(num\_A\_dot[zone\_e\_dot] - num\_A\_dot[zone\_e\_dot + 1]);

u3 = 1 - u4;

}

Double u13 = (u1 \* u3), u23 = (u2 \* u3), u14 = (u1 \* u4), u24 = (u2 \* u4);

Double u = ((u13 \* get\_table\_A(zone\_e\_dot, zone\_e) + u23 \* get\_table\_A(zone\_e\_dot, zone\_e + 1) + u14 \* get\_table\_A(zone\_e\_dot + 1, zone\_e) + u24 \* get\_table\_A(zone\_e\_dot + 1, zone\_e + 1)) / (u13 + u23 + u14 + u24));

u = u + u\_previos;

if (u >= 10.0) u = 10.0;

else if (u <= -10.0) u = -10.0;

//if (u < 0.0) u = u \* -1.0;

return u;

}

public static Double CalculateNextY(Double y\_previos, Double u\_previos, Double num)

{

return (1.01 \* y\_previos + 0.01 \* Math.Pow(y\_previos, 2.0) + num \* u\_previos);

}

public static Double CalculateNextY(Double x)

{

return x;

}

public static Double CalculateNextK\_A(Double k)

{

if (0 <= k && k <= 200) return 1;

else if (201 <= k && k <= 400) return 2;

else return -1;

}

public static Double CalculateNextK\_B(Double k)

{

return Math.Sin(2 \* Math.PI \* k / 200.0);

}

}

private void System1(Chart chartY)

{

Double[] D = new Double[MAX];

Double[] num = { 0.03, 0.02, 0.01 };

Color[] colors= { Color.Blue, Color.Green,Color.Red,Color.Gray };

for (int i = 0; i < 600; i++)

{

D[i + 1] = Fuzzy.CalculateNextK\_A(i);

}

//標題 最大數值

Series series\_D = new Series("D", 1);

//設定線條顏色

series\_D.Color = Color.Yellow;

//折線圖

series\_D.ChartType = SeriesChartType.Line;

//將數值新增至序列

for (int index = 0; index < MAX; index++)

{

series\_D.Points.AddXY(index, D[index]);

}

//將序列新增到圖上

chart\_out.Series.Add(series\_D);

for (int index = 0; index < num.Length; index++)

{

int max = 0;

Double[] Y = new Double[MAX];

Double[] U = new Double[MAX];

E[] e = new E[MAX];

e[0] = new E(0, 0);

Y[0] = 0;

U[0] = 0;

Double[] E\_ = new Double[MAX];

Double[] E\_dot = new Double[MAX];

E\_[0] = e[0].e;

E\_dot[0] = e[0].e\_dot;

for (int i = 0; i < 600; i++)

{

try

{

e[i + 1] = Fuzzy.FuzzyControl\_A(i, Y[i], e[i].e);

U[i + 1] = Fuzzy.CalculateU\_A(e[i + 1].e, e[i + 1].e\_dot, U[i]);

Y[i + 1] = Fuzzy.CalculateNextY(Y[i], U[i + 1], num[index]);

E\_[i] = e[i].e;

E\_dot[i] = e[i].e\_dot;

max = i;

if (Y[i + 1] >= 2.4) {

i = -1;

Fuzzy.Single\_pole[1] -= 0.1;

Fuzzy.Single\_pole[3] += 0.1;

textBox1.Text = "";

foreach (double pole in Fuzzy.Single\_pole)

{

textBox1.Text += pole + " ";

}

continue;

//break;

}

}

catch(Exception ex)

{

break;

}

}

//標題 最大數值

Series series1 = new Series(num[index].ToString(), 1);

//設定線條顏色

series1.Color = colors[index];

//折線圖

series1.ChartType = SeriesChartType.Line;

//將數值新增至序列

for (int x = 0; x < max; x++)

{

series1.Points.AddXY(x, Y[x]);

}

//將序列新增到圖上

chartY.Series.Add(series1);

//series1.IsValueShownAsLabel = true;

//series2.IsValueShownAsLabel = true;

}

//標題

chartY.Titles.Add("System1");

}

}