Abstract: This paper presents a simple new algorithm for smoothing curves in Computer Graphics based on concepts from Fuzzy Logic. The algorithm is generalized to handle manipulable smooth curves as required in Computer Graphics design-work. The result is reminiscent of some descriptions of Fuzzy Expert Systems and so this leads into a graphical analysis of and comparison with standard Fuzzy Expert Systems. The analysis shows that fuzzy smoothed polylines can provide a better controller response curve than the standard methods of Fuzzy Expert Systems do currently.

Key-Words: fuzzy expert system, computer graphics, curve drawing

1. Introduction
This paper presents a graphical analysis of the performance of a widely accepted standard Fuzzy Expert Systems (FES) [4]. If a FES is going to control the acceleration and deceleration of a train, tram or lift carrying human passengers, or control the movement of robots and milling machinery in an environment where humans are also working and so forth it behoves us to be able to see the response curve of the system (before we use it) so as to gain confidence about the overall performance of the FES. As well as being a good approximation to the requirements of a system, a FES should not have any sudden or jerking motions. The response curve of the FES should not only match the system requirements for which it is a controller but should also exhibit smoothness. In graphical analysis presented below, the response curves of a standard FES is shown to have non-smoothness as a common occurrence.

Humans prefer gradual transitions from one state to another rather than sudden changes. With slow transitions underway we can prepare ourselves or adapt our existing plan for the coming changes. On the other hand sudden changes are likely to cause accidents and damage when humans are in an environment where unexpected system responses occur and where the changes occur quicker than the time it takes for humans to comfortably adjust to the changes. In the first part of this paper we will look at the application of this principle in fuzzy set theory to smoothing polylines in Computer Graphics. This results in a visual smooth curve editing algorithm that allows us to rapidly design shapes to order in Computer Graphics. The algorithm compares favourably against existing techniques for shaping smooth curves and has the advantages of simpler mathematics and intuitive visual control.

In the second part of this paper we do a graphical and mathematical analysis of typical Fuzzy Expert Systems. For simplicity we are concerned here with single (real valued) input single (real valued) output systems and we analyse the effect of the configuration of input and output macrostates on the plot of the response curve of the FES. In doing this we are looking for rules that allow us to set up the macrostate configurations in order to get a desired target response curve. This is similar task to the Computer Graphics problem as discussed in part one of the paper and this then leads to a detailed comparison of the two approaches.

2 Smoothed Curves
We live in an analog world and yet we make digital models of it. Over time our modelling is refined to better and better levels of approximation to the analog world. Discrete modelling however involves sudden sharp state changes which do not occur in the real analog world. For example, a car modelled to have only two speeds : maximum and zero is not a practical or palatable reality – we need far more shades of variation for a comfortable ride. Similarly a car that can make perfect 90 degree turns without slowing down is not mechanically possible in the real world. Consider a car travelling at speed \( v \) along the x-axis and which at time \( t^* \) changes to motion parallel to the y-axis at speed \( v \). The velocity profiles of this motion (Fig 1a) have unrealistic transitions which involve infinite decelerations and infinite accelerations. Nature abhors such infinities and we also prefer gradual transitions. Fuzzy theorists insist that we cannot know
velocities to infinite precision as implied in this description and that precise discrete transitions are not realistic. Fuzzy set theory has typically used a linear ramp to implement such real-world transitions as shown in Fig 1b. If we plot the motion of the car based on the linear ramping of fuzzy set theory (Fig 1b) then we get the smooth curve displayed in Fig 2a. The car is travelling initially with velocity \( u = (v,0) \) until time \( t = t_1 = t* - T/2 \) where \( T \) is the total transition time. From time \( t_1 \) to \( t_2 = t* + T/2 \) the x-component of velocity \( v_x \) linearly reduces to 0 while at the same time the y-component of velocity \( v_y \) linearly increases from 0 to \( v \). After time \( t_2 \) the car continues along the y-axis with velocity \( w = (0,v) \).

When this process is modelled digitally great care has to be taken to avoid overshoot and undershoot. If the transition period \( T \) is broken down into \( n \) steps then we have:

\[
\begin{align*}
 v_{x,i} &= u_x + \frac{i}{n} \Delta v_x \\
 v_{y,i} &= u_y + \frac{i}{n} \Delta v_y \\
 t_i &= t_1 + \frac{i}{n} T
\end{align*}
\]

for \( i = 0 \) to \( n \) where \( u_x = v, u_y = 0, \Delta v_x = -v \) and \( \Delta v_y = v \). The car’s position during the transition period is given by

\[
\begin{align*}
 x_{i+1} &= x_i + v_{x,i} \Delta t \\
 y_{i+1} &= y_i + v_{y,i} \Delta t \\
 \Delta t &= T/n
\end{align*}
\]

These equations however result in overshoot in comparison with the case \( T = 0 \). The cause of this discrepancy is precisely the problem of discrete modelling of analog systems. From the recurrence relations above the endpoint \( x_n = (x_n,y_n) \) is:

\[
 x_n = x_o + nu \Delta t + (n-1) \Delta v \Delta t
\]

In contrast the analog solution is

\[
 x_n = x_o + n w \Delta t
\]

where \( w = u + \Delta v \) is the final linear velocity after the turn.

The method adopted for getting the discrete model back into synchronization with the real model was to replace \( n \) with \( n-1 \) in formula (1). This means that the \( n \)th velocity overshoots its target of \( w \) but we don’t need \( v_n \) anyway and so the final position is exactly on target.

Furthermore when we want to generate smooth curves with no straight sections the velocity vectors \( u \) and \( w \) are rescaled so that their sum equals the displacement from start to endpoint of the curve i.e \( x_n - x_0 \).

This algorithm enables us to easily make and edit arbitrary curves by specifying any number of points that the curve must pass through and the required curve tangents at those points. All curve editing is totally local and unlike the usual spline curves used for shape generation the curve is guaranteed to pass through the given points. A curve editor program was written to demonstrate this. The program allows the user to digitise any number of points (called control points) through which the curve must pass and for each such point the user generates an arrow which specifies the desired tangent at that point. (See Fig 2b.) At any time the user can select a control point and parallel move the arrow to any other position or reset the endpoint of the tangent arrow at that control point to point in a different direction. The resultant curve is continuous and smooth. Restrictions on generating the curve are that adjacent arrows may not be parallel (or antiparallel) and that the starting point of the second must be in the half-plane whose edge passes through the starting point of the first arrow and containing the arrow as the normal to the half-plane’s edge and otherwise unrestricted. The equations of the curves connecting arrows is:

\[
x(t) = x_0 + at + bt^2
\]

where

\[
a = u - \frac{1}{2n(n-1)}(w - u)
\]

and

\[
b = \frac{1}{2(n-1)T}(w - u)
\]

Thus the curve is essentially a piecewise parametric quadratic in contrast to the usual piecewise cubic splines. This means for instance that the curves can be multivalued as in closed shapes. The curves can approximate circles and ellipses. A piecewise quadratic spline curve was proposed in [3]. The equations for the curve sections were a subcase of (3) where \( y \) was quadratic in \( x \). Special coding was required to avoid cusps and in order to ensure smoothness between sections local control was sacrificed. Although for a small number of control points the curve was well-behaved it became too oscillatory for higher numbers of control points. However the algorithm described above
has none of these drawbacks and additionally is much simpler to implement. We will now look at Fuzzy Expert Systems as see how this algorithm relates to them.

3 Fuzzy Expert System Response Curves

The purpose of an FES is to provide an approximation of the behaviour of a complex system that is good enough to allow the system to be controlled. A system with multiple inputs and multiple outputs over finite real ranges can be modelled by another system containing only a single real input x and a single real output y. Experts familiar with such a system may not know the functional relationship \( y = f(x) \) for the system but may nevertheless be able to describe its nature fairly accurately over a number of patches on \((x,y)\) space. With Fuzzy Expert Systems we specify a number of relevant ranges (i.e. input macrostates) along the x-axis and likewise a number of other relevant ranges or output macrostates along the y-axis. The fuzzy rules in the FES simply state that if \( x \) is in a particular input macrostate then the system would find \( y \) in a particular output macrostate. These rules result in rectangular patches in \((x,y)\) space through which the curve \( y = f(x) \) passes. The macrostates are actually fuzzy sets rather than real intervals. The macrostate memberships are usually represented as isosceles triangles whose base is the corresponding variable range. To compute the crisp output \( y \) for a given crisp input \( x \) the FES determines the membership of input \( x \) in every input macrostate and looks up the corresponding fuzzy rule. In Bart Kosko’s type of FES \([1,5 & \text{cf. 4}]\) the input membership scales down the corresponding output macrostate membership function and then all resulting memberships are simply added (at the same \( x \) values) to form the FES output fuzzy set (even though this could result in memberships exceeding unity). This resultant fuzzy set is defuzzified by the centroid method to yield the expected output \( y \) \([4,5]\).

Although this is not the first or only FES algorithm variant we will start with this FES algorithm to analyse the kinds of functions that this sort of FES can generate. Suppose that there are \( m \) input states \( i = 1 \) to \( m \) with centres \( a_i \) and spreads \( b_i \) for \( i = 1 \) to \( m \). (Therefore the base of fuzzy input state \( i \) ranges from \( a_i - b_i/2 \) to \( a_i + b_i/2 \).) Likewise suppose that there are \( n \) output states with centres \( c_j \) and spreads \( d_j \) for \( i = 1 \) to \( n \). There should be at least \( r = m \) rules for otherwise there would be unused input macrostates. The simplest FES would have \( m = 1, n = 1 \) and \( r = 1 \) rule that input macrostate 1 maps to output macrostate 1. The response curve in this case is simply \( y = f(x) = c_1 \). More generally, if there are \( m > 0 \) input macrostates and they are non-overlapping then the response curve \( f(x) \) is a step function taking the constant value \( c_j \) over the corresponding input macrostate range. It makes no difference whether the output macrostates are overlapping or not, non-overlapping input states generates nothing more than a simple step function as shown in Figures 3a and 3b. Next consider an FES with \( m = 2, b_1 = b_2 \) and \( |a_1 - a_2| < b_1/2 \). This as illustrated in Fig 4 results in a step function but with a linear ramp from one step value to the other. In general if all input macrostates have equal spreads then the response curve \( f(x) \) is equal to the corresponding \( c_j \) value inside the non-overlapping parts of the input macrostates with linear connecting ramps within the overlapping parts of the input macrostates. This picture is not changed if the output macrostates are overlapping. The picture is also unchanged if more than one input macrostate is mapped to the same output macrostate.

Up to this point, the FES has not generated a curve of any great interest that could not be generated more simply in other ways and these curves are additionally non-smooth and not very helpful in approximating complex system behaviour functions. The FES response curve gets more interesting when there is overlap and the spreads are not equal. Consider the simple case \( m = 2 \) with \( a_1 < a_2 \) and \( b_2 = 2b_1 \), \( n = 2 \) with input macrostate 1 mapping to output macrostate 1 and input macrostate 2 mapping to output macrostate 2. This FES shows the usual flat sections in the regions where the input macrostates do not overlap. However in the region where the input macrostates do overlap the connection is concave downwards as shown in Fig 5. In the reverse condition where \( b_2 = 2b_1 \) we find that the connection between flat sections is concave upwards in the overlap region. What is the nature of this curve? Analysis shows that in the overlap region the response curve \( f(x) \) is the ratio of two first order functions of \( x \):

\[
f(x) = \frac{A\mu_1 + B\mu_2}{C\mu_1 + D\mu_2}
\]

where

\[
\mu_1 = \frac{2}{b_1}(a_1 + \frac{b_1}{2} - x)
\]

\[
\mu_2 = \frac{2}{b_2}(x - a_2 + \frac{b_2}{2})
\]

Fig 4 results in a step function but with a linear ramp from one step value to the other. In general if all input macrostates have equal spreads then the response curve \( f(x) \) is equal to the corresponding \( c_j \) value inside the non-overlapping parts of the input macrostates with linear connecting ramps within the overlapping parts of the input macrostates. This picture is not changed if the output macrostates are overlapping. The picture is also unchanged if more than one input macrostate is mapped to the same output macrostate.
and A, B, C and D are constants built from \(c_1, d_1, c_2\) and \(d_2\). Thus the functional form of the curve pieces produced is linear rational.

So to summarize, we can use an FES to approximate a real system behaviour curve by appropriately piecing together overlapping patches in which \(f(x)\) is rational linear. The parameters of this curve are dependent largely on the relative widths of the overlapping patches. Since in real systems we don’t want flat sections, there must be no unoverlapped patches and patches should be of different widths. A question that arises then is whether it is easier to approximate arbitrary curves in this way by designing overlapping rectangular patches scaled relatively in the x-direction to give the appropriate curvature or in the new way described in part 1 of this paper. From writing and using programs to generate curves in both ways I have found that the arrow control method of part 1 generates target curves much more easily than the standard FES approach of part 2.

4 Conclusions
Fuzzy Expert Systems can produce a wide variety of response curves. By doing a graphical (and mathematical) analysis of FES cases we can discern the higher level rules by which FESs generate curves. We observed that curve smoothness requires that no FES rule patches should be unoverlapped. We also observed the manner in which the overlapped patch curves are formed and concluded that generating complex behavioural functions \(y = f(x)\) by this method was somewhat difficult. Only double triangle overlaps were analysed mathematically. Triple triangle overlaps were observed graphically to produce more complex curve shapes. The standard graphical methods of cubic splines would seem an easier way of controlling curve shape and the new method introduced in part 1 of this paper is even easier than that.

References