A Graphical User Interface for Analysis of Friction Stir Welds

Michael Janes
Mathematics and CS
SDSM&T
Rapid City, SD 57701
Michael.Janes@sdsmt.edu

Enkhsaikhan Boldsaikhan
National Institute for
Aviation Research
Wichita State Univ.
Wichita, KS 67260
eboldsaikhan@niar.wichita.edu

Edward Corwin
Antonette Logar
Mathematics and CS
SDSM&T
Rapid City, SD 57701
Antonette.Logar@sdsmt.edu

ABSTRACT

This work describes the use of splines in displaying the phase space diagram of a Friction Stir Weld. One of the research areas in the Center for Friction Stir Processing is the development of a technique for evaluating weld quality in a non-destructive way, that is, the flaws must be apparent without having to cut the weld and inspect it, and it must be fast enough to run in real-time. One production idea is a video-game-like interface with an input device that will allow an operator to adjust the weld parameters by responding to visual queues generated during welding. One of these visual queues is the shape of the phase space diagram. The phase space diagram is a plot of Y Forces (sideways movement) against their time derivatives. Using splines to generate the plot produces an easy to analyze image.

1 INTRODUCTION

Friction stir welding is an innovative technique for joining metals that was developed and patented by The Welding Institute of Cambridge, England in 1991 [1]. The process of friction stir welding (FSW) begins by plunging a rotating pin tool in-between two work pieces (Figure 1). The rotation of the pin tool produces two important side effects. First it generates enough heat to plasticize the work pieces, and second, it stirs the two work pieces together.

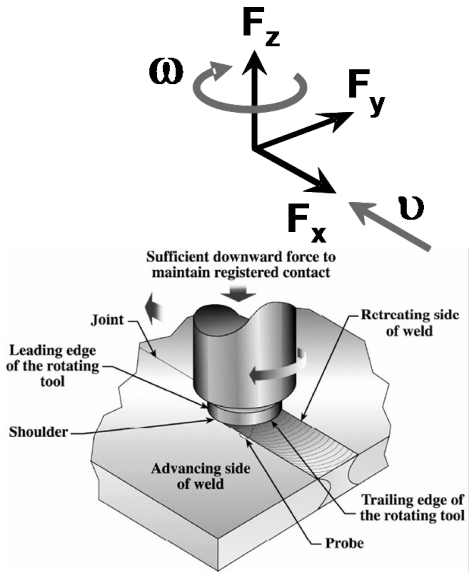

Figure 1: Drawing of a typical FSW pin tool

While welding, the operator must define three important parameters: the pin tools revolutions per minute, the pin tools forward speed, and the downward force exerted on the pin tool. An improper selection of these parameters may result in a weld with undesirable defects. The goal of this research is to find a non-destructive way to analyze the weld process and tweak the system parameters to produce the optimum weld.

The ISTIR-10 located in the AMP Center at SDSM&T is capable of recording a variety of feedback forces, three of which were found to be useful in analyzing a weld: the X, Y, and Z feedback forces. The X force measures the resistance to the forward motion of the pin tool, the Z force measures of the up and down pressure exerted on the pin tool, and the Y force measures the amount of sideways movement made during the weld (Figure 1). The stability of the Y force, in particular, has been shown to be highly correlated with weld quality. A phase space diagram was found to be an appropriate method for visualizing the stability of the Y force.

A phase space diagram is a dynamical systems analysis method that plots a parameter against its derivative (in this case the Y force against its derivative). In an ideal world the phase plot would produce a single stationary circle. The amount of divergence from this stationary circle can be used to measure weld quality. A phase plot for a good weld is displayed on the left and a phase plot for a bad weld is displayed on the right.

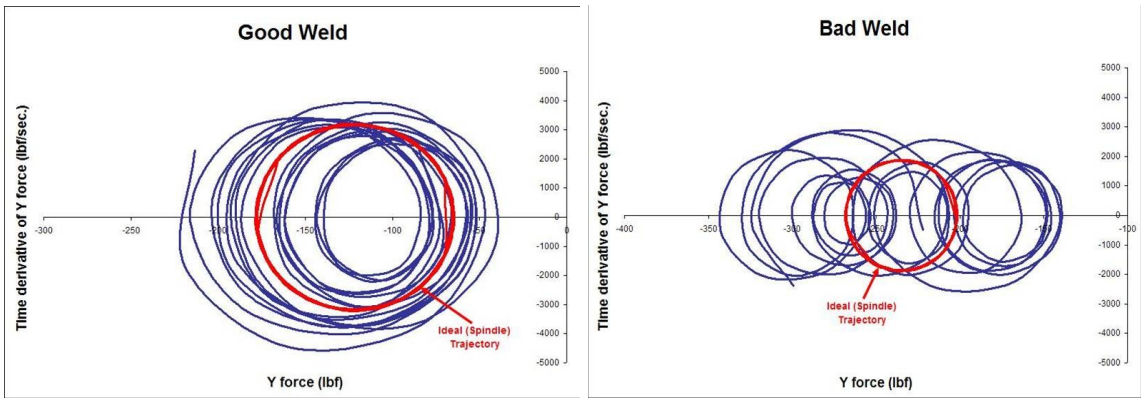

Figure 2: Phase plot of a good weld (left) and a bad weld (right). Notice the stationary circular pattern of the good weld compared to the roaming pattern of the bad weld [1].

1.2 FITTING A CURVE

The objective was to come up with a curve fitting technique that accurately displays the phase space diagram. The ultimate goal of the FSW application is to produce a control system that can: receive feedback during the welding process, be able to analyze the feedback, and correct the welder if necessary. One approach is a video-game-like interface with an input device that will allow an operator to adjust the weld parameters by responding to visual queues generated during welding. If the operator is to perform his or her task the data needs to be presented in an accurate, and easy to analyze manner. In the case of the phase space diagrams the plots of Y forces against their derivatives need to be fitted with curves to recognize the desired circular trajectory.

2 CHOOSING A SPLINE

Microsoft Windows GDI+ supports two types of curve fitting techniques Cardinal splines and Bezier splines. Both splines are powerful and useful graphical devises. The following sections give a detailed description of both Cardinal splines and Bezier splines.

2.1 CARDINAL SPLINES

Cardinal splines are a subset of cubic Hermite splines. A cubic Hermite spline is a third degree piecewise parameterized curve, that is, there exists n - 3 third degree polynomials that define the curves between n vertices (first and last vertices are not included). The problem of defining a Hermite spline becomes the problem of finding the coefficients for the n – 3 third degree polynomials. Equation 1 shows the parameterized curve that must be derived for each vertex.

$$x = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y = a_y t^3 + b_y t^2 + c_y t + d_y$$

Equation1: Third degree parameterized curve defining the x coordinate (top), and the y coordinate (bottom), t is on the interval (0 to 1) [4].

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix}$$

Equation 2: Matrix from of Equation1 [4].

To solve for the coefficients two constraints are enforced:

1) The parameterized curve intersects all inner vertices.

   A Cardinal spline is an interpolating spline, meaning that it passes through all given vertices. The Cardinal spline presented here will only intersect inner vertices, that is the first and last vertices will not be included in the spline. The method used to calculate the slope of the curve at any vertex uses the two adjacent vertices – (n-1 and n+1). Therefore, the first and last vertices cannot be included in the spline, but are used only to calculate the slope of the curve. Different approximation techniques can be used to include the first and last vertex if necessary.

2) The derivatives of two connecting parameterized curves match

   The derivative at the end of a parameterized curve must equal the derivative at the start of its connecting parameterized curve. ($x_n'(1) = x_{n+1}'(0)$, and $y_n'(1) = y_{n+1}'(0)$). Equation 3 shows the derivative of the desired parameterized curve in matrix form.

$$\begin{bmatrix} \frac{dx}{dt} & \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix}$$

Equation 3: Derivative of the parameterized curve [4].

By applying the two mentioned constraints eight equations are formed to go along with the eight coefficients. Equation 4 shows the eight equations in matrix form.

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \frac{dx_1}{dt} & \frac{dy_1}{dt} \\ \frac{dx_2}{dt} & \frac{dy_2}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix}$$

Equation 4: In the right coefficient matrix are the coefficients that define the parameterized curve between point (x_1, y_1) and point (x_2, y_2). Note that the derivative at both points most be provided to solve for the coefficients [4].

Solving Equation 4 for the coefficient matrix by multiplying both sides of the equal sign by the inverse of the 4 by 4 matrix gives the resulting Hermite spline equation.

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Hermite}} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \frac{dx_1}{dt} & \frac{dy_1}{dt} \\ \frac{dx_2}{dt} & \frac{dy_2}{dt} \end{bmatrix}}_{\mathbf{G}_{Hermite}}$$

Equation 5: Final Hermite spline equation [4].

Notice that to solve for the coefficients the derivatives of both vertices (x_1, y_1) and (x_2, y_2) must be found. A Cardinal spline is just a special case of a Hermite spline that uses Equation 6 in place of each vertex's derivative. The equations use the constant t, the tension parameter, to help define the smoothness of the curve.

$$dx_n/dt = t * (x_{n+1} - x_{n-1})$$
$$dy_n/dt = t * (y_{n+1} - y_{n-1})$$

Equation 6: Equations that are used in place of derivatives for a Cardinal spline [4].

The tension parameter is usually a value between 0 and 1. A tension of 0 corresponds to infinite tension, forcing the curve to take the shortest path between vertices (straight lines). A tension of 1 corresponds to no tension, allowing the spline to take the path of least total bend. With any tension value greater than 1, the curve behaves like a compressed spring, pushed to take a longer path [2]. Figure 2 shows the effect of different tension values.

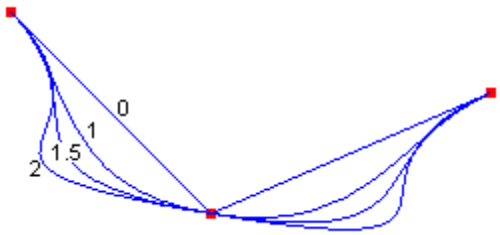

Figure 2: Four Cardinal splines with four different tension values [2]

2.2 BEZIER SPLINES

Like Cardinal splines, cubic Bezier splines can be represented in Hermite matrix form. Unlike Cardinal splines, the generated third degree, parameterized curve does not intersect all inner points. Bezier splines are approximating splines, their curves are designed to curve toward inner points but not necessarily intersect them. The basis behind a cubic Bezier spline is that for a set of four points: the first and last points are anchor points, that is, they intersect the curve, and the two middle points are control points, which control the curve by defining the tangent to the curve. Figure 3 shows a Bezier spline between four points (There can be any number of control points, but cubic Bezier spline, by definition, has 2 control points).

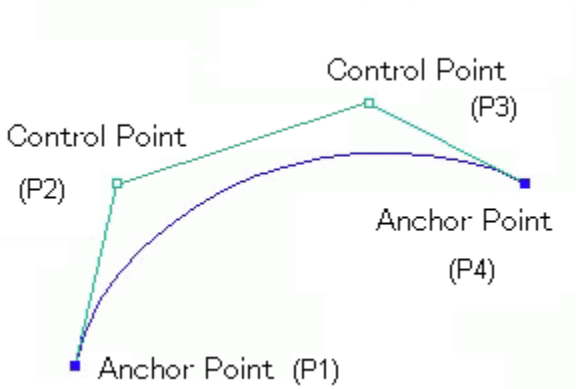

Figure 3: Bezier spline between four points. Notice that only two points intersect the curve (these points are known as anchor points) [3].

The gradients of the anchor vertices can be expressed in terms of the adjacent control vertex and itself. Equation 7 shows the gradients of the anchor vertices.

$$\nabla p_1 = 3(p_2 - p_1)$$
$$\nabla p_4 = 3(p_4 - p_3)$$

Equation 7: Slope of the curve at the two anchor vertices – p_1 and p_4 [4].

Again the problem becomes finding the eight coefficients that define the curve between the four vertices. Using Equation 7 that defines the slope of the curve a Bezier spline can be formulated in terms of a Hermite matrix like the one in Equation 8.

$$\underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \frac{dx_1}{dt} & \frac{dy_1}{dt} \\ \frac{dx_2}{dt} & \frac{dy_2}{dt} \end{bmatrix}}_{\mathbf{G}_{Hermite}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

Equation 8: Bezier spline matrix. Notice that points (x_1, y_1) and (x_2, y_2) in the matrix $G_{Hermite}$ are equal to points (x_1, x_1) and (x_4, y_4) in the matrix G_{Bezier} respectively, these are the anchor points. Also notice that the control points (x_2, y_2) and (x_3, and y_3) in the G_{Bezier} matrix are only used to find the two derivatives [4].

Substituting the right side of Equation 8 for $G_{Hermite}$ in Equation 5 gives Equation 9.

$$\begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Hermite}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

Equation 9: Bezier matrix substituted into the Hermite matrix equation [4].

Multiply the matrices in Equation 9 to get Equation 10, which are the eight equations for the eight unknown polynomial coefficients.

$$\begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \underbrace{\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Bezier}} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

Equation 10: Bezier form of a Hermite matrix [4].

Another method used to calculate a Bezier spline is Bernstein Bezier formulation. The Bernstein Bezier formulation is based on the subdivision property of Bezier splines. To construct the subdivision draw lines connecting all the vertices together (except the two anchor vertices), and then recursively draw lines between ratios of distances (if, in Figure 4, P_0^1 is located at $1/10^{th}$ the distance between P_0 and P_1 then P_1^1 needs to be located at $1/10^{th}$ the distance between P_1 and P_2) for a total of n-2 iterations, where n is the degree of the Bezier curve (3 in the case of a cubic spline). After n-2 iterations a vertex on the curve will be found. To construct the entire curve simply start the ratio at zero, run the subdivision routine, place a pixel on that vertex, and repeat incrementing the ratio by 1 / (number of vertices on the curve) for each iteration.

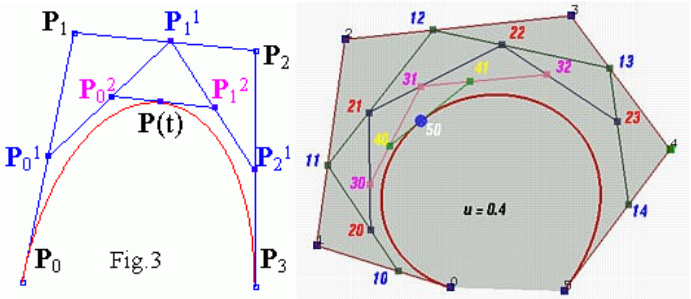

Figure 4: Bernstein Bezier formulation for cubic Bezier curve (left), and a Bernstein Bezier formulation for a sixth degree Bezier curve (right) [6].

To define a Bezier spline for more than 4 vertices the Bernstein Bezier formulation can be used to create a Bezier spline of degree n – 1, where n is the number of vertices (Figure 4 right), or a series of Bezier splines can be spliced together creating a piecewise curve. The problem with splicing Bezier splines together is that there is no guarantee that the tangents to the two connecting curves will equal each other at the connected endpoint (like a Cardinal spline). If the tangents do not equal each other the composite Bezier curve will look jagged and unsmooth. To smooth out the composite curve, control vertices are added where needed to equalize the tangents.

2.3 COMPARISON OF CARDINAL SPLINES AND BEZIER SPLINES

Both Cardinal splines and Bezier splines are fast and easy to implement, but no spline is right for every situation. The following two lists give the basic properties of first Cardinal splines then Bezier Splines.

Properties of Cardinal splines:

1) The curve is a C^1 – smooth curve [7].

   C^n referrers to a continuity number, which is a measure of how smoothly connected a piecewise parameterized curve is. For a curve to have C^0

continuity means that the piecewise curves are connected at their endpoints. C^1 continuity guarantees that that the first derivatives of two connected curves are equal at the connected endpoint. Likewise C^n continuity guarantees that the n^{th} derivatives of the connected two curves are equal at the connected endpoint. Note that C^n continuity implies C^{n-1}, C^{n-2} . . . and C^0 continuity.

2) The curve interpolates all but the first and last vertices.

   Note that when using the Microsoft Windows GDI+ built in DrawCurve method (which draws a Cardinal spline interpolation of a given array of vertices at a given tension) derivative approximations are made to include the first and last vertex in the interpolation.

3) When a vertex is changed only part of the curve needs to be recalculated [7].

   If vertex P^3 was changed then the curves with right end vertices of P^2, P^3, P^4, and P^5 would need to be recalculated.

4) The piecewise parameterized curve is affine invariant [7].

   A curve is said to be Affine Invariant if the coordinate system it is represented in can change without affecting the relative geometry of the curve. This can be seen by the fact that the geometry of the curve remains consistent when the curve is rotated, scaled, or translated.

5) Defining a Cardinal spline is linear with respect to the number of vertices.

   To define a Cardinal spline the coefficients for n- 3 parameterized curves must be found. Matrix multiplication is the time dominating operation in finding the coefficients, and since the matrix multiplication is always done on two matrices that have fixed sizes, each part of the piecewise curve can be found in constant time, making the entire algorithm linear.

6) The tension parameter can be changed to alter the amount of curve.

Properties of Bezier splines:

1) Bezier curves lie in the convex hull generated by its control and anchor vertices [7].

   This fact could be helpful in finding out if two curves intersect. If the two convex hulls do not intersect the curves do not intersect.

2) A properly combined composite Bezier curve has C^1 continuity [7].

To properly combine Bezier curves control vertices must be added or moved to insure the tangents to two connecting curves are equal at the connecting endpoint. Note that Microsoft Windows GDI+'s DrawBeziers method produces a Bezier curve with only C^0 continuity, and no smoothing at connection points.

3) When a vertex is changed on a composite Bezier curve only part of the curve needs to be recalculated [7].

   If vertex P^3 was changed then the curves with right end vertices of P^3, and P^4 would need to be recalculated.

4) Defining a composite cubic Bezier curve is linear.

   To define a composite cubic Bezier curve the coefficients for n/3 parameterized curves must be found. Like Cardinal splines matrix multiplication is the time dominating operation, and can be done in constant time, making the algorithm linear.

5) The degree of the Bezier curve is directly related to the number of control points it contains.

6) Composite Bezier curves are affine invariant [7].

7) Bezier splines are approximating splines.

After comparing both splines, A Cardinal spline is the obvious choice for the FSW application. Since Cardinal splines are interpolating splines, they will accurately display the trajectories. Also the tension parameter on a Cardinal spline can be calibrated to produce the optimal curve, and patterns in the phase space diagram will be more easily detected. With a cubic Bezier spline the trajectories will appear distorted since for any set of four data points only two points intersect to the curve. Also, Microsoft's DrawBeziers function can not be used for this application since it produces a spline with only C^0 continuity; the image would appear pieced together and unsmooth. A Bezier spline with C^1 continuity could be produced, but finding a C^1 Bezier spline is far more complicated than defining a Cardinal spline.

4 IMPLEMENTATION

The implementation of Cardinal splines in the FSW application uses Microsoft Windows GDI+ DrawCurve method. The DrawCurve method takes three parameters: a pen object, an array of points, and a tension parameter. The pen object is the object that defines how the curve is drawn (color, width of line, etc.). The array of points must be an array of point objects (either integer value points – Point, or floating-point value points -

PointF). The tension parameter is optional, if it is not supplied a default value of .5 is used.

The time it takes to display an array of N points using the DrawCurve method is on the order of $O(N^3)$. Note that this is an approximation found by fitting a curve to N plotted against the time it took to display the curve interpolation of the N points. Although $O(N^3)$ is relatively high the DrawCurve method can display the one to two hundred points needed to analyze a section of weld in real time.

Figures 5 and 6 are screen shots of the FSW application's phase space diagram. Both Figures are displaying the same data, with Figure 5 using lines to connect the data vertices and Figure 6 using a Cardinal spline interpolation of the data vertices.

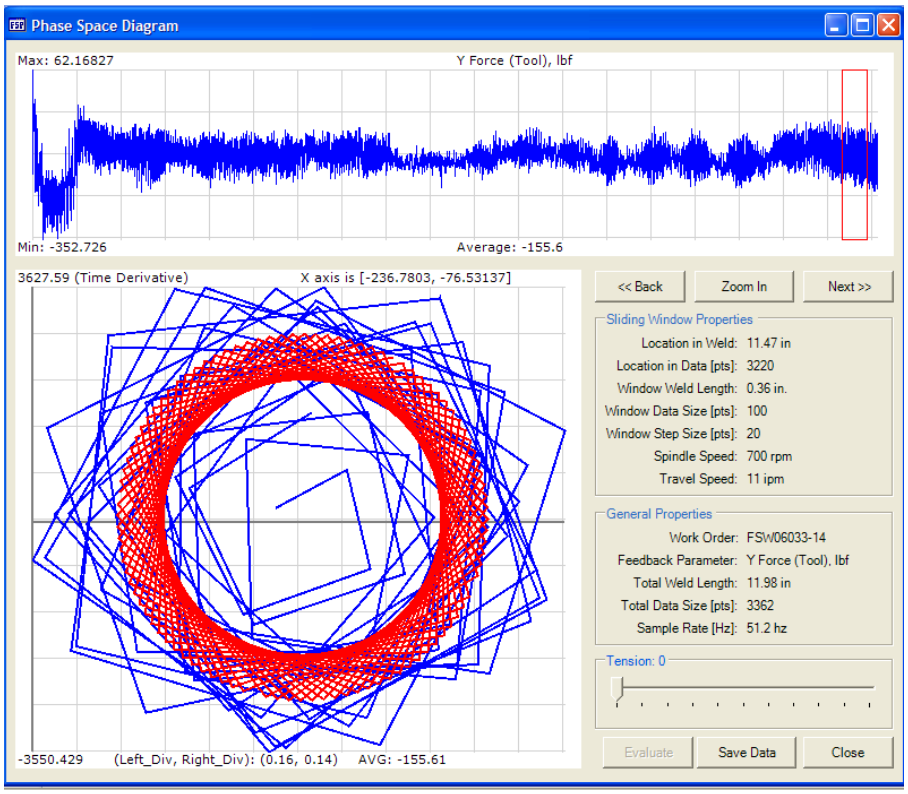

Figure 5: Phase space diagram using lines to connect data points.

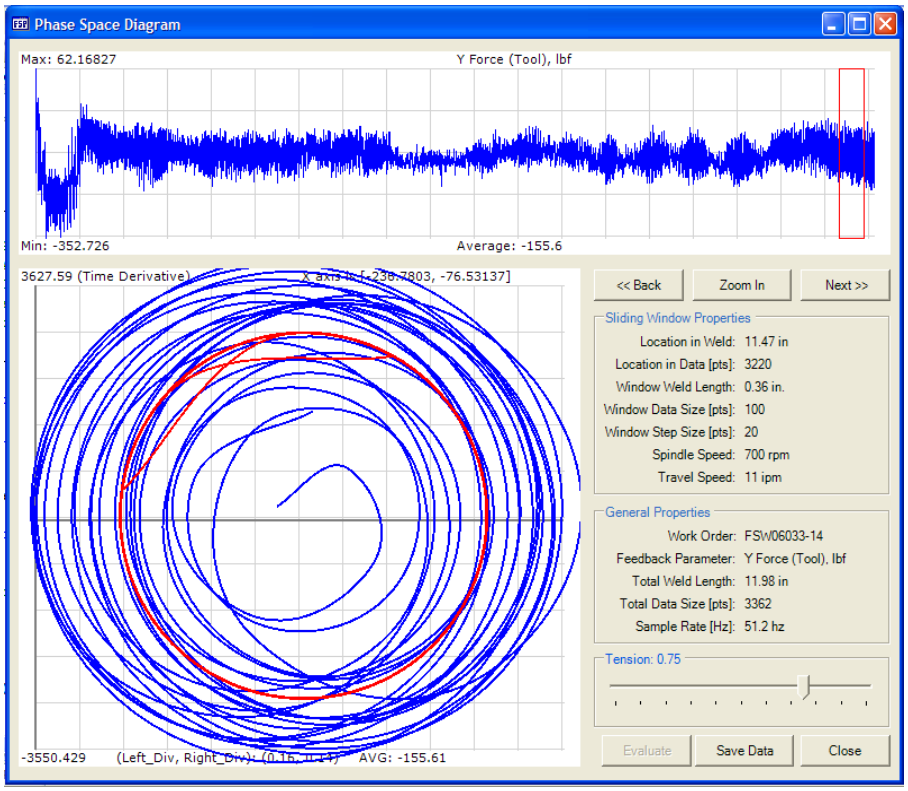

Figure 6: Phase space diagram using Cardinal spline interpolation.

Notice that Figure 5 has a clear circular pattern, but it is unclear to what extent the circles wander along the horizontal axis, while Figure 6 clearly shows the range of deviation from the desired center circle. The amount of deviation from the center circle is the most important factor when using the phase space diagram to analyze a weld. Clearly, using spline interpolations aids the user in analyzing the phase space diagram, and ultimately in producing a better weld.

5 CONCLUSION

The goal was to select a curve fitting technique that accurately, and in real time, displays a phase space diagram. It has been concluded that between a Bezier spline and a Cardinal spline, the Cardinal spline is the best choice to meet this goal. Cardinal splines not only quickly and accurately display a spline interpolation, they also provide the added flexibility of an adjustable tension parameter. This work examined two types of splines but many others exist. However, the two selected are commonly used representatives of interpolating and approximating splines. Descriptions of different splines can be found in countless books and websites, some of these books and websites are provided in the reference section.

References

[1] Boldsaikhan, E., Corwin, E. M., Logar, A. M., McGough, J., and Arbegast, W. J., "Phase Space Analysis of Friction Stir Weld Quality," Friction Stir Welding and Processing IV, Edited by R.S. Mishra, et al, TMS, 2007.

[2] Cardinal Splines. Retrieved February 25, 2009, from Microsoft MSDN Web site: http://msdn.microsoft.com/en-us/library/ms536358.aspx

[3] Drawing Bezier Splines. Retrieved February 25, 2009, from Microsoft MSDN Web site: http://msdn.microsoft.com/en-us/library/ms533926(VS.85).aspx

[4] Fisher, H., (2004, April 4th). "Spline." Retrieved February 25, 2009, from Australian National University Web site: http://escience.anu.edu.au/lecture/cg/Spline/tableOfContent.en.html

[5] Glaeser, G., "Fast Algorithms for3D-Graphics." New York: Springer-Verlag, 1994.

[6] Panne, M., (1996). "Parametric Curves." Retrieved February 25, 2009, from University of Toronto Web site: http://www.cs.helsinki.fi/group/goa/mallinnus/curves/curves.html

[7] Plis, A., Shikin, E., "Handbook on Splines for the User." New York: CRC Press Inc., 1995.

[8] Pipenbrinck, N., (1998, March 30th). "Hermite Curve Interpolation". Retrieved February 25, 2009, Web site: http://www.cubic.org/docs/hermite.htm