Finding the Shortest but Smooth Path for the Path of a Robot.

kaw@eng.usf.edu , http://numericalmethods.eng.usf.edu/mws

NOTE: This worksheet demonstrates the use of Maple for finding the shortest but smooth path for the path of a robot in the area of manufacturing. It illustrates how spline interpolation can be used to determine this path.

Introduction

The following example illustrates the real world use of interpolation to find the shortest but smooth path of a robot. A robot arm equipped with a laser is doing a quick quality check of the radius on six holes on a rectangular plate 15" x 10". The center locations of the six holes are given as (2, 7.2), (4.25, 7.1), (5.25, 6), (7.81, 5), (9.2, 3.5), (10.6, 5). Find the shortest but smooth path for the robot from the first to the last point.

```
> restart;
```

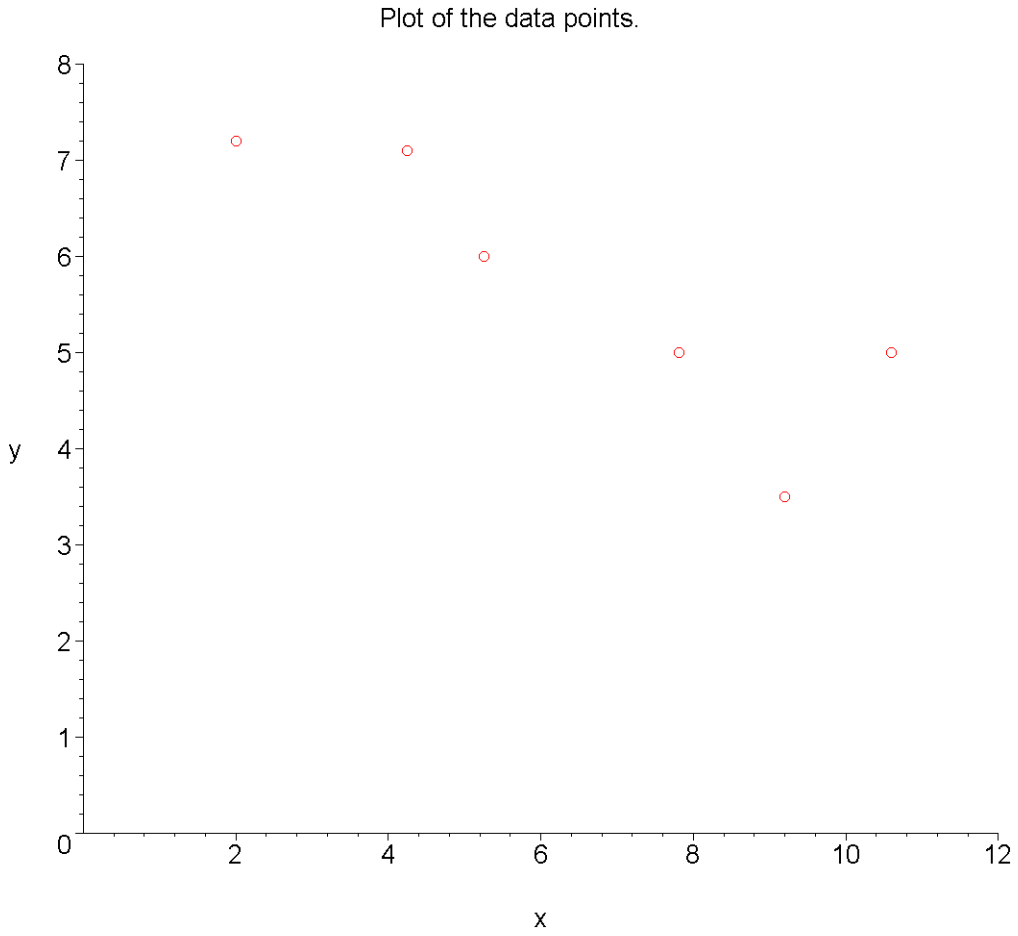
Section I : Data.

The following is the data (x-y) coordinate data of the center of the six holes.

```
> xy:=[[2,7.2],[4.25,7.1],[5.25,6],[7.81,5],[9.2,3.5],[10.6,5]]:
```

Plotting the data:

```
> plot(xy,x=0..12,y=0..8,style=POINT,symbol=CIRCLE,symbolsize=20,tit
  le="Plot of the data points.");
```



Section II: Linear Splines.

Connecting the consecutive data points using linear splines will give the shortest path. However, this path is not smooth as the derivatives will be discontinuous at the interior data points. However, this will form a baseline for other calculations in the next two sections.

```
> linear_spline:=spline([2,4.25,5.25,7.81,9.2,10.6],[7.2,7.1,6,5,
  3.5,5],x,linear);
```

$$linear\_spline := \begin{cases} 7.288888889 - 0.04444444444\,x & x < 4.25 \\ 11.77500000 - 1.100000000\,x & x < 5.25 \\ 8.050781250 - 0.3906250000\,x & x < 7.81 \\ 13.42805755 - 1.079136691\,x & x < 9.2 \\ -6.357142857 + 1.071428571\,x & otherwise \end{cases}$$

The length S of a curve 'y' from 'a' to 'b' is given by $S = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$.

```
> a:=2:
> b:=10.6:
```

The length of the linear spline, 'Slinear' is :

```
> Slinear:= int((1+diff(linear_spline,x)^2)^0.5,x=a..b);
```

$$Slinear := 10.58405613$$

Section III: Polynomial Interpolation.

Since the robot has to pass through six data points, we can interpolate the data to a fifth order polynomial. Since a fifth order polynomial has continuous first derivatives, the path given by the polynomial would be smooth. The fifth order polynomial is,

```
> polynomial_function:=interp([2,4.25,5.25,7.81,9.2,10.6],[7.2,7.
  1,6,5,3.5,5],x);
```

$$polynomial\_function := 41.34437571\,x - 30.89819894 - 15.85478362\,x^2 + 2.786231103\,x^3 - 0.2309138595\,x^4 + 0.007292341221\,x^5$$

The length of the polynomial function, 'Spoly' is :

```
> Spoly:= int((1+diff(polynomial_function,x)^2)^0.5,x=a..b);
```

$$Spoly := 13.12335423$$

Section IV: Spline Interpolation.

Clearly the length of the curve from the polynomial interpolation is larger than the length obtained from linear spline interpolation. Let us use cubic spline interpolation. Since a cubic spline interpolant has continuous first derivative, the path given by it would also be smooth.

```
> cubic_spline:=spline([2,4.25,5.25,7.81,9.2,10.6],[7.2,7.1,6,5,3
  .5,5],x,cubic);
```

$$cubic\_spline := \begin{cases} 6.412057670 + 0.393971164900000003\,x - 0.394745964311166748\,10^{-15}\,(x-2)^2 - 0.0866006141700000066\,(x-2)^3, & x < 4.25 \\ 11.01542157 - 0.921275662999999967\,x - 0.584554145698898852\,(-4.25+x)^2 + 0.405829808700000016\,(-4.25+x)^3, & x < 5.25 \\ 10.58269627 - 0.872894528300000028\,x + 0.632935280042843162\,(-5.25+x)^2 - 0.173651853599999989\,(-5.25+x)^3, & x < 7.81 \\ 13.17238603 - 1.04640025999999998\,x - 0.700710956330525270\,(-7.81+x)^2 + 0.487165155999999988\,(-7.81+x)^3, & x < 9.2 \end{cases}$$

$$5.069716181 - 0.170621324000000019\,x + 1.33076774467731740\,(-9.2+x)^2 - 0.316849463100000006\,(-9.2+x)^3\ ,\ otherwise$$

The length of the cubic spline, 'Scubic' is :

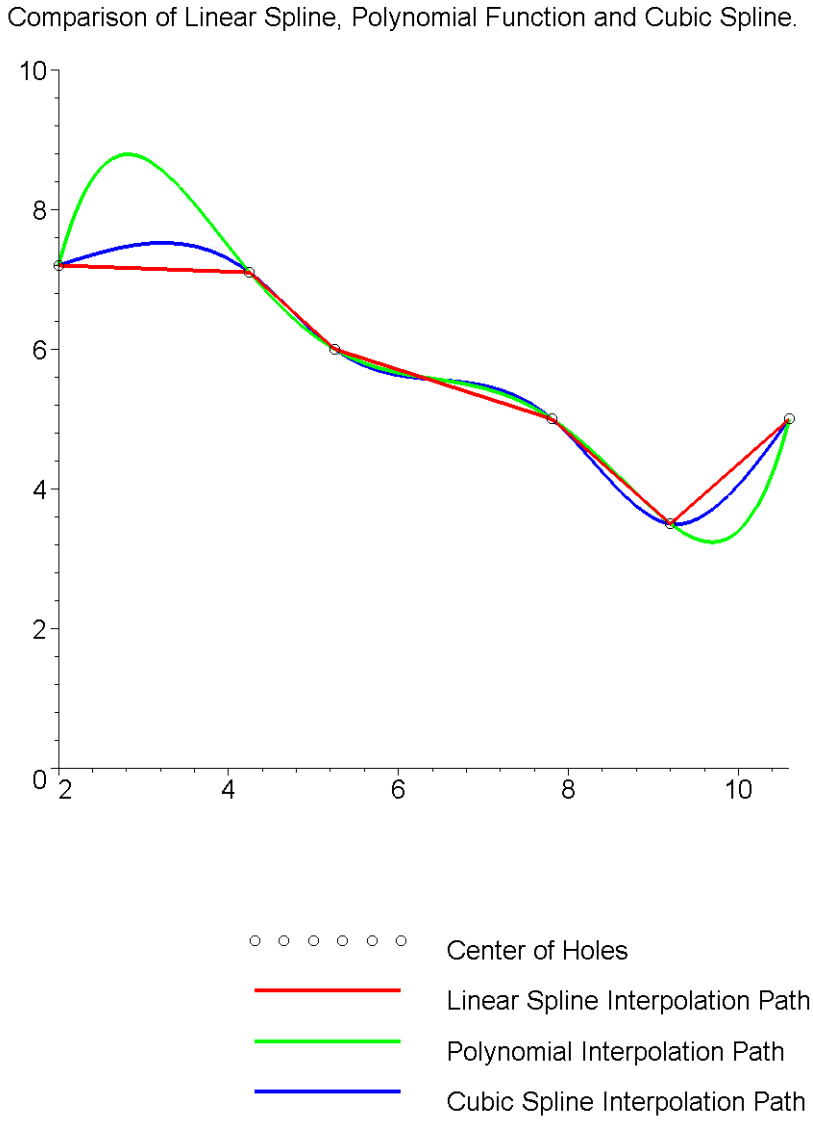
```
> Scubic:= int((1+diff(cubic_spline,x)^2)^0.5,x=a..b);
```

$$Scubic := 10.92884812$$

Section V: Comparison.

Below is a plot to compare the paths obtained using linear splines, polynomial function and cubic spline :

```
> linear_spline:=x->spline([2,4.25,5.25,7.81,9.2,10.6],[7.2,7.1,6
  ,5,3.5,5],x,linear):
> polynomial_function:=x->interp([2,4.25,5.25,7.81,9.2,10.6],[7.2
  ,7.1,6,5,3.5,5],x):
> cubic_spline:=x->spline([2,4.25,5.25,7.81,9.2,10.6],[7.2,7.1,6,
  5,3.5,5],x,cubic):
> plot([xy,linear_spline,polynomial_function,cubic_spline],2..10.
  6,0..10,style=[POINT,LINE,LINE,LINE],symbol=CIRCLE,symbolsize=2
  0,thickness=4,title="Comparison of Linear Spline, Polynomial
  Function and Cubic
  Spline.",color=[BLACK,RED,GREEN,BLUE],legend=["Center of
  Holes","Linear Spline Interpolation Path", "Polynomial
  Interpolation Path","Cubic Spline Interpolation Path"]);
```



>

>

>

Section VI: Conclusion.

Maple helped us to apply our knowledge of numerical methods of interpolation to find the shortest but smooth path of the robot. Using Maple functions and plotting routines made it easy to find a solution efficiently.

Can you check the length of the path by using Quadratic Splines? Is the path shorter than what we obtained using cubic splines and fifth order polynomial interpolation?

References

[1] *Autar Kaw and Michael Keteltas, Holistic Numerical Methods Institute, See http://numericalmethods.eng.usf.edu/mws/ind/05inp/mws_ind_inp_spe_shortestpath.pdf*