Topic : Additional Interpolation Topics
Simulation : Choosing the shortest path for a robot
Language : Mathematica 4.1
Authors : Nathan Collier,Autar Kaw
Date : 12 December 2002
Abstract : A rapid robot arm with a laser is used to make seven holes on a rectangular plate 12" X 8" at six points as shown. The path of the robot going from one indentation point to another needs to be smooth so as to avoid sharp jerks in the arm that can otherwise create premature wear and tear of the robot arm. One suggestion has been to fit a fifth order polynomial through the six points. Another suggestion was to fit a quadratic spline through the holes. Which suggestion is better so that the robot path is shorter but also smooth?

In[218]:=
```
ClearAll;
```

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

Array of x data

In[219]:=
```
x := {2, 4.5, 5.25, 7.81, 9.2, 10.6}
```

Array of y data

In[220]:=
```
y := {7.2, 7.1, 6, 5, 3.5, 5}
```

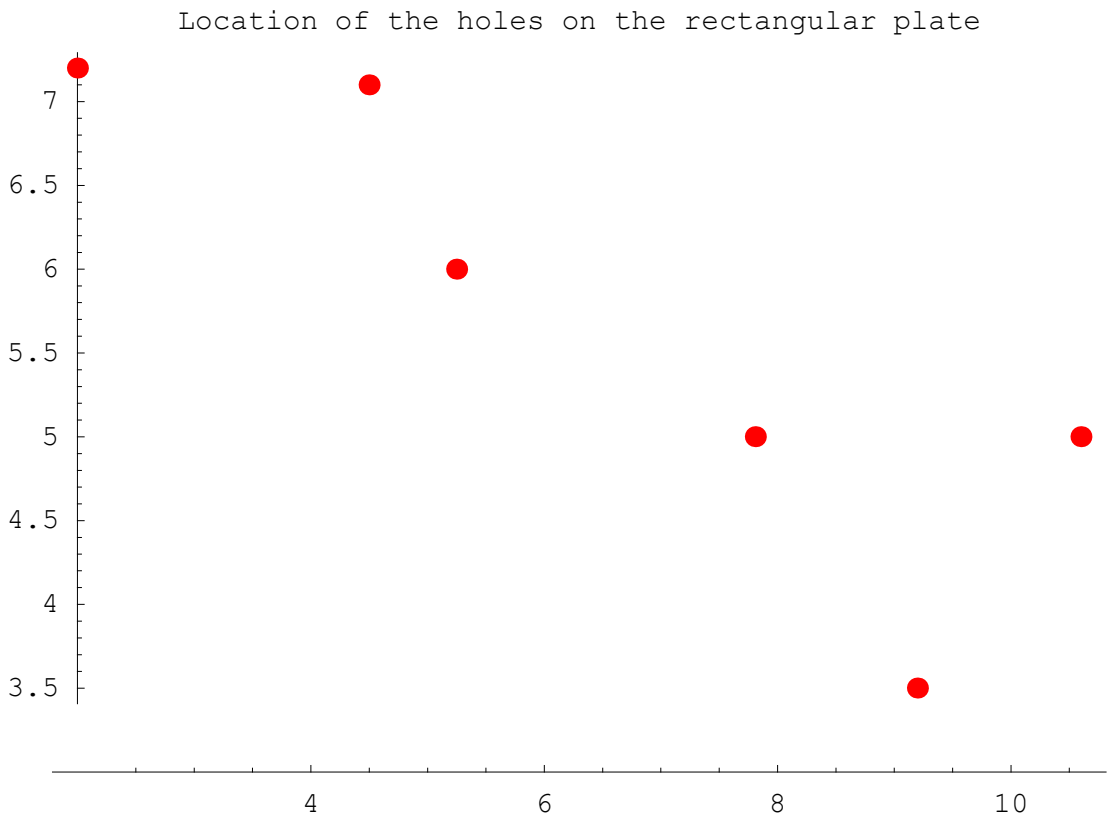
■ SOLUTION

In[221]:=
```
n := Abs[Dimensions[x]][[1]]
```

In[222]:=
```
xy = Table[0, {i, n}, {j, 2}];
```

In[223]:=
```
Do[xy[[i, 1]] = x[[i]]; xy[[i, 2]] = y[[i]], {i, 1, n}]
```

In[224]:=
```
data = ListPlot[xy, PlotStyle → {Red, PointSize[0.02]},
    PlotLabel -> "Location of the holes on the rectangular plate",
    TextStyle → {FontSize → 11}, AxesOrigin → {2, 3}];
```



Polynomial Interpolation

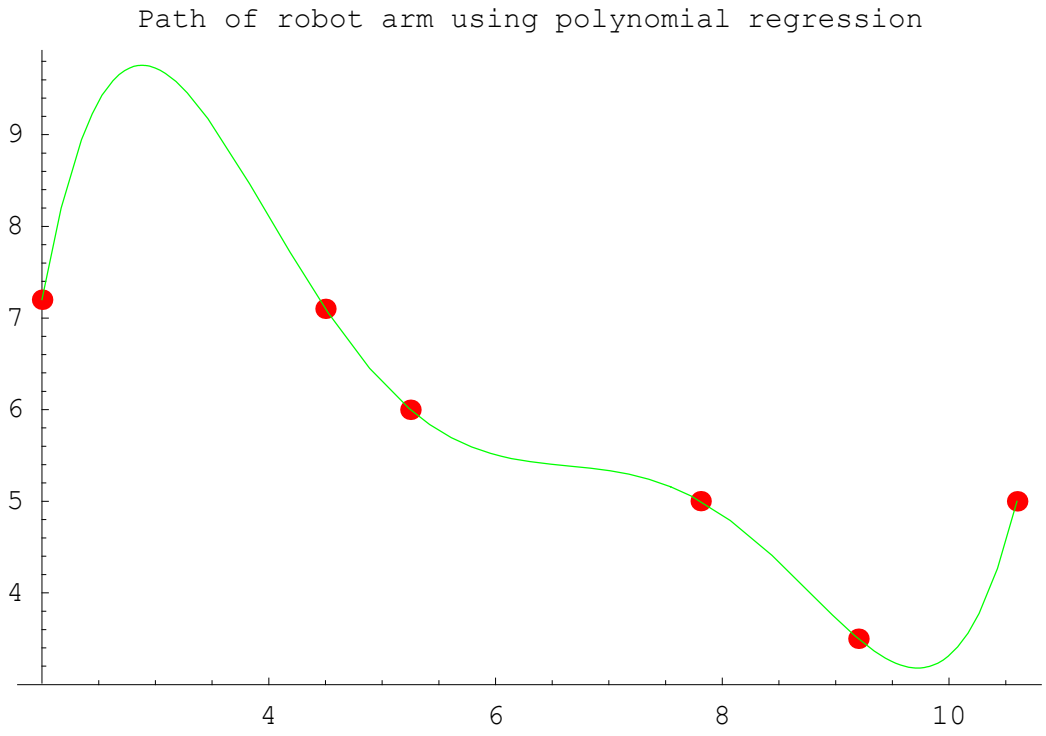
Using polynomial, find a path that goes through the six data points

In[225]:=
```
M = Table[x[[i + 1]]^j, {i, 0, n - 1}, {j, 0, n - 1}];
A = LinearSolve[M, y];
```

In[227]:=
```
fp[z_] := Sum[A[[i]] * z^(i - 1), {i, 1, n}]
```

In[228]:=
```
plot1 =
  Plot[fp[z], {z, 2, 10.6}, PlotStyle → {Green}, DisplayFunction → Identity];
```

In[229]:=
```
Show[data, plot1,
  PlotLabel -> "Path of robot arm using polynomial regression"];
```



Length of curve

In[230]:=
```
Lp = 0;
Do[Lp = Lp + ((fp[i] - fp[i - 0.01])^2 + 0.01^2)^0.5, {i, 2.01, 10.6, 0.01}]
```

In[232]:=
```
Lp
```

Out[232]=
```
14.9193
```

Cubic Spline Interpolation

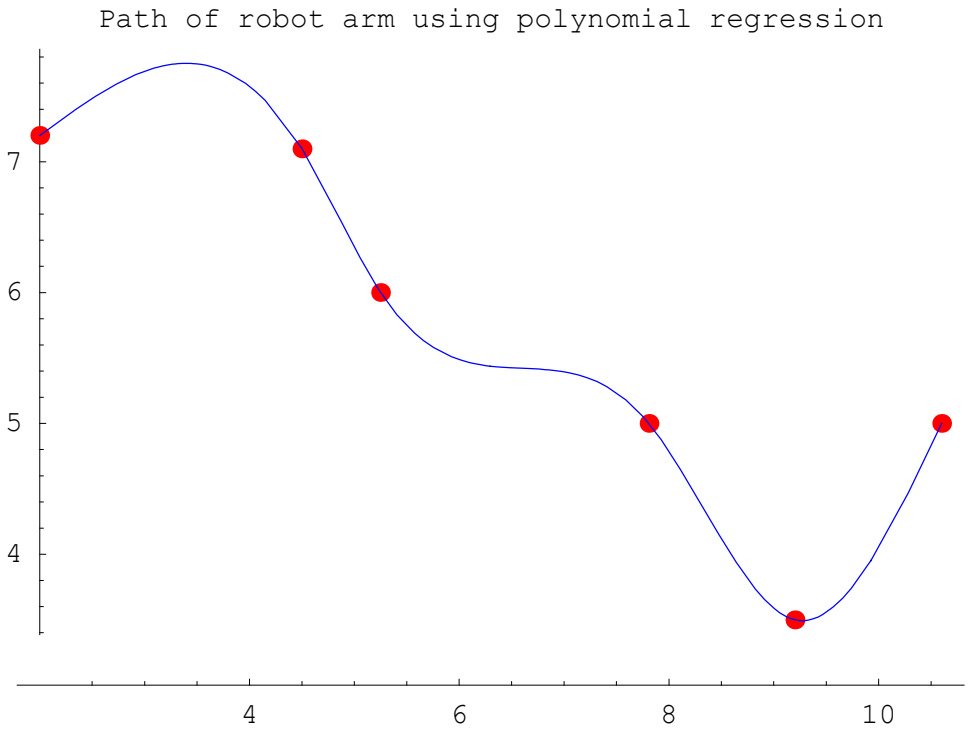
Using cubic splines, find a path that goes through the six data points. *Mathematica*'s Cubic Spline function works in a parametric fashion. The user specifies the relative position in the data set, and the function interpolates based on that. For more information, see the help file. For this reason, we must write our own cubic spline process. The following was adapted from an algorithm found in Burden's *Numerical Analysis*, 4th ed. page 131.

In[233]:=
```
CubicSplineCoeff[X0_, Y0_] := Module[{X = X0, Y = Y0},
    GettingCoeff := Module[{},
      n := Length[X] - 1;
      a = b = d = Alpha = h = Table[0, {n}];
      l = u = z = c = Table[0, {n + 1}];
      Do[a[[i]] = Y[[i]], {i, 1, n}];
      Do[h[[i]] = X[[i + 1]] - X[[i]], {i, 1, n}];
      Do[Alpha[[i]] = 3 * (Y[[i + 1]] * h[[i - 1]] - Y[[i]] * (X[[i + 1]] - X[[i - 1]]) +
            Y[[i - 1]] * h[[i]]) / (h[[i - 1]] * h[[i]]), {i, 2, n}];
      l[[1]] := 1;
      u[[1]] := 0;
      z[[1]] := 0;
      Do[
       l[[i]] = 2 * (X[[i + 1]] - X[[i - 1]]) - h[[i - 1]] * u[[i - 1]];
       u[[i]] = h[[i]] / l[[i]];
       z[[i]] = (Alpha[[i]] - h[[i - 1]] * z[[i - 1]]) / l[[i]],
       {i, 2, n}];
      l[[n + 1]] := 1;
      z[[n + 1]] := 0;
      c[[n + 1]] := 0;
      j = n + 1; While[(j = j - 1) ≥ 1,
       c[[j]] = z[[j]] - u[[j]] * c[[j + 1]];
       b[[j]] = (Y[[j + 1]] - Y[[j]]) / h[[j]] - h[[j]] * (c[[j + 1]] + 2 * c[[j]]) / 3;
       d[[j]] = (c[[j + 1]] - c[[j]]) / (3 * h[[j]]);];];
    SplineFunction[t_] := Module[{i},
      For[i = 1, i ≤ n, i++,
       If[X[[i]] ≤ t && t < X[[i + 1]], k = i];];
      If[t < X[[1]], k = 1];
      If[t > X[[n + 1]], k = n];
      q = t - X[[k]];
      Return[((d[[k]] * q + c[[k]]) * q + b[[k]]) * q + a[[k]]];];
    GettingCoeff;];
```

In[234]:=
```
CubicSplineCoeff[x, y]
```

In[235]:=
```
plot2 = Plot[SplineFunction[z], {z, 2, 10.6},
    PlotStyle → {Blue}, DisplayFunction → Identity];
```

In[236]:=
```
Show[data, plot2,
  PlotLabel -> "Path of robot arm using polynomial regression"];
```



Length of curve

In[237]:=
```
Ls = 0;
Do[Ls = Ls + ((SplineFunction[i] - SplineFunction[i - 0.01]) ^2 + 0.01^2) ^0.5,
 {i, 2.01, 10.6, 0.01}]
```

In[239]:=
```
Ls
```

Out[239]=
```
11.2476
```

Compare the two curves

In[240]:= **Show[data, plot1, plot2];**