Spline Method of Interpolation--Simulation

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

NOTE: This worksheet demonstrates the use of Maple to illustrate the spline method of interpolation. We limit this worksheet to linear and quadratic spline interpolation.

Introduction

The Spline method of interpolation (for detailed explanation, you can read the textbook notes and examples, and see a Power Point Presentation) is illustrated. Given 'n' data points of 'y' versus 'x', it is required to find the value of y at a particular value of x using linear and quadratic splines.

```
> restart;
> with(LinearAlgebra):
  with(linalg):
Warning, the previous binding of the name GramSchmidt has been removed and it
now has an assigned value

Warning, the protected names norm and trace have been redefined and
unprotected
```

Section I : Data.

The following is the array of x-y data which is used to interpolate. It is obtained from the physical problem of velocity of rocket (y-values) vs. time (x-values) data. We are asked to find the velocity at an intermediate point of x=16.

```
> xy:=[[10,227.04],[0,0],[20,517.35],[15,362.78],[30,901.67],[22.
  5,602.97]]:
```

Value of x at which y is desired

```
> xdesired:=16:
```

Section II : Big scary functions.

This function will sort the data matrix into ascending order and puts them into a new matrix.

```
> n:=rowdim(xy):
> for passnum from 1 to n-1 do
    for i from 1 to n-passnum do
      if xy[i,1]>xy[i+1,1] then
        temp1:=xy[i,1];
        temp2:=xy[i,2];
```

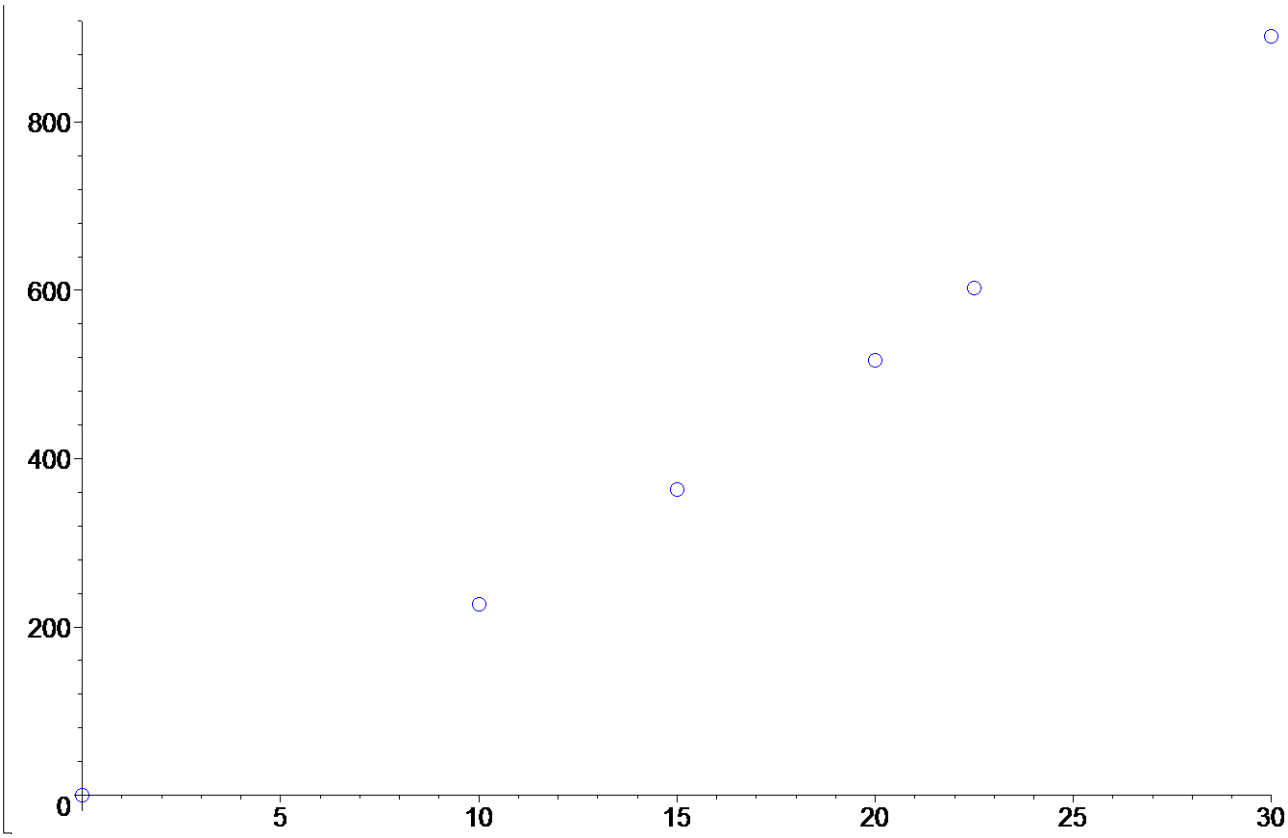
```
        xy[i,1]:=xy[i+1,1];
        xy[i,2]:=xy[i+1,2];
        xy[i+1,1]:=temp1;
        xy[i+1,2]:=temp2;
      end if:
    end do:
  end do:
```

```
> x:=matrix(n,1):
  y:=matrix(n,1):
  for i from 1 to n do
  x[i,1]:=xy[i,1];
  y[i,1]:=xy[i,2];
  end do:
```

```
> ranger:=proc(ar,n)
  local i,xmin,xmax,xrange;
  xmin:=ar[1,1]:
  xmax:=ar[1,1]:
  for i from 1 to n do
  if ar[i,1] > xmax then xmax:=ar[i,1] end if;
  if ar[i,1] < xmin then xmin:=ar[i,1] end if;
  end do;
  xrange:=xmin..xmax;
  end proc:
```

Plotting the given values of X and Y.

```
> plot(xy,ranger(x,n),style=POINT,color=BLUE,symbol=CIRCLE,symbol
  size=30);
>
```



Section III: Linear spline interpolation

Given $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_{n-1}, y_{n-1})(x_n, y_n)$, fit linear splines to the data. This simply involves forming the consecutive data through straight lines. So the data is sorted into an ascending order; the linear splines are given by $(y_i = f(x_i))$

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0), \qquad x_0 \le x \le x_1$$

$$= f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1), \qquad x_1 \le x \le x_2$$

$$\ldots\ldots\ldots$$

$$= f(x_{n-1}) + \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x - x_{n-1}), \qquad x_{n-1} \le x \le x_n$$

```
> flinear:=proc(z)
  local d,i;
  if z < x[2,1] then
  d:=y[2,1]+(y[2,1]-y[1,1])/(x[2,1]-x[1,1])*(z-x[2,1]);
  else
  for i from 2 to n do
    if (z > x[i,1]) and (z <= x[i+1,1]) then
      d:=y[i,1]+(y[i+1,1]-y[i,1])/(x[i+1,1]-x[i,1])*(z-x[i,1]);
    end if;
  end do;
```

```
  end if;
  d;
  end proc:
```

Value of function at desired value of X is

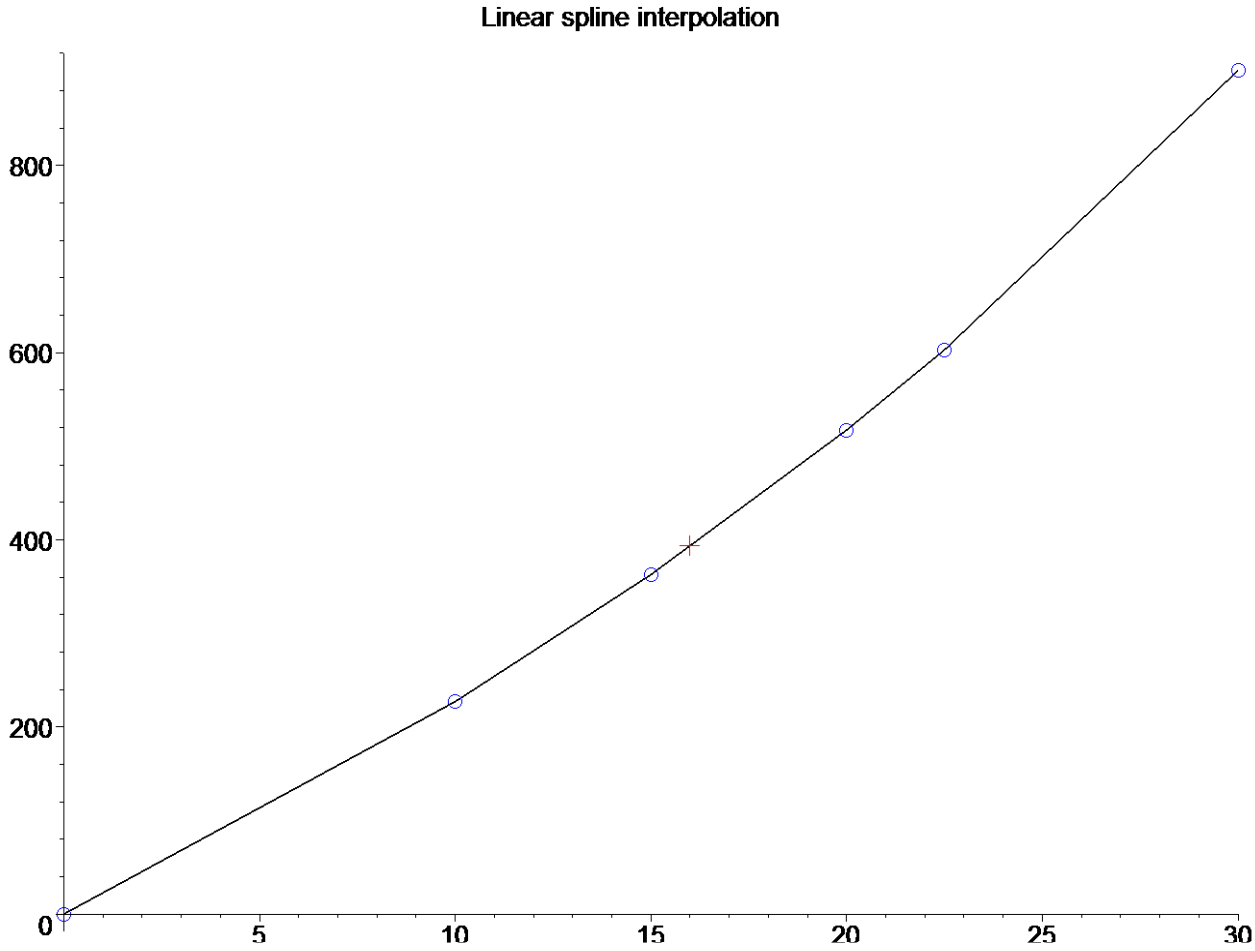
```
> flinear(xdesired);
```

393.6940000

```
> fprev:=%:
```

Plotting the linear spline interpolant and the value of Y for the desired X

```
> plot([xy,[[xdesired,flinear(xdesired)]],flinear],ranger(x,n),st
  yle=[POINT,POINT,LINE],color=[BLUE,RED,BLACK],symbol=[CIRCLE,CR
  OSS],symbolsize=[30,40],thickness=2,title="Linear spline
  interpolation");
```



Section IV: Quadratic interpolation

In these splines, a quadratic polynomial approximates the data between two consecutive data points. Given $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, fit quadratic splines through the data. The splines are given by

$$f(x) = a_1 x^2 + b_1 x + c_1,\ x_0 \le x \le x_1$$

$= a_2 x^2 + b_2 x + c_2,\ x_1 \le x \le x_2$

.....

$= a_n x^2 + b_n x + c_n,\ x_{n-1} \le x \le x_n$

There are 3n such coefficients a_i, $i = 1, 2, \ldots, n$; b_i, $i = 1, 2, \ldots, n$; c_i, $i = 1, 2, \ldots, n$

To find '3n' unknowns, one needs to set up '3n' equations and then simultaneously solve them. To know more about how to setup these '3n' equations, please click on textbook notes. After setting up these equations, they are solved using the matrix method. The following function assembles the matrix whose inverse is needed to solve for the coefficients of the polynomial splines that fits the data.

```
> A:=matrix(3*(n-1),3*(n-1),0):
> for i from 0 to 3*(n-1)-1 do
    for j from 0 to 3*(n-1)-1 do
      A[i+1,j+1]:=0;
    end do;
  end do;

  for i from 1 to n-1 do
    for j from 0 to 1 do
      for k from 0 to 2 do
        A[2*i-1+j+1,3*i-3+k+1]:=x[i+j,1]^k;
      end do;
    end do;
  end do;

  for i from 1 to n-2 do
    for j from 0 to 1 do
      for k from 0 to 1 do
        A[2*(n-1)+i+1,3*i-2+k+j*3+1]:=(-1)^j*(2*x[i+1,1])^k;
      end do;
    end do;
  end do;

  A[1,3]:=1:
```

The following gives the matrix which is required to solve for the coefficients of the quadratic splines

```
evalm(A);
```

$$\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 10 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 10 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 15 & 225 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 15 & 225 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 20 & 400 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 20 & 400 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 22.5 & 506.25 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 22.5 & 506.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 30 & 900 \\
0 & 1 & 20 & 0 & -1 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 30 & 0 & -1 & -30 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 40 & 0 & -1 & -40 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 45.0 & 0 & -1. & -45.0
\end{bmatrix}$$

This assembles the Y matrix also needed to determine the coefficients of the quadratic splines.

```
> Y:=matrix(3*(n-1),1,0):
  for i from 0 to n-2 do
  for j from 0 to 1 do
  Y[2*(i+1)+j,1]:=y[i+j+1,1];
  end do;
  end do;
  evalm(Y);
```

$$\begin{bmatrix}
0 \\
0 \\
227.04 \\
227.04 \\
362.78 \\
362.78 \\
517.35 \\
517.35 \\
602.97 \\
602.97 \\
901.67 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}$$

Solving for the coefficients, we get

```
> C:=evalm(inverse(A) &* Y);
```

$$C := \begin{bmatrix} 0. \\ 22.70400000 \\ 0.1054334397\ 10^{-13} \\ 88.880000 \\ 4.9280000 \\ 0.88880000 \\ -141.610000 \\ 35.6600000 \\ -0.13560000 \\ 554.55000 \\ -33.956000 \\ 1.60480000 \\ -152.13 \\ 28.8600000 \\ 0.20888888 \end{bmatrix}$$

```
> fquad:=proc(z)
  local d,i,j:
  if z <= x[2,1] then
    d:=C[1,1]+C[2,1]*z+C[3,1]*z^2:
  else
    for i from 2 to n-1 do
      if z <= x[i+1,1] and z > x[i,1] then
        d:=0:
        for j from 0 to 2 do
          d:=d+C[3*(i-1)+j+1,1]*z^j:
        end do:
      end if:
    end do:
  end if:
  d;
  end proc:
```

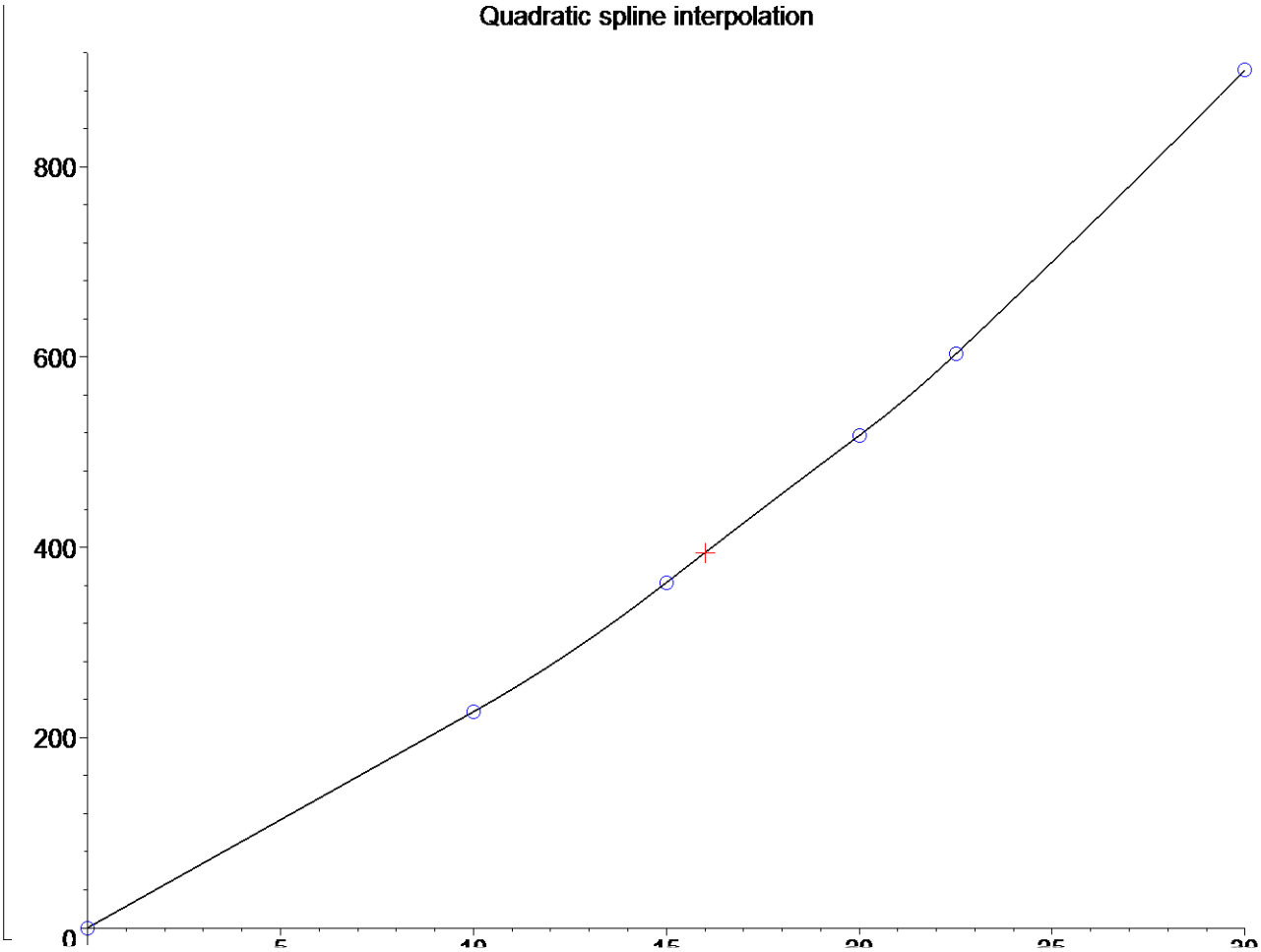
Value of function at desired value of X is

```
> fquad(xdesired);
```

394.2364000

Plotting the quadratic spline interpolant and the value of Y for the desired X

```
> plot([xy,[[xdesired,fquad(xdesired)]],fquad],ranger(x,n),style=
  [POINT,POINT,LINE],color=[BLUE,RED,BLACK],symbol=[CIRCLE,CROSS]
  ,symbolsize=[30,40],thickness=2,title="Quadratic spline
  interpolation");
```



Section V: Cubic spline interpolation

The algorithm of cubic spline interpolation is not shown. However, we are using the Maple function to conduct the cubic spline interpolation,

```
> fcubic:=t->spline(convert(x,vector),convert(y,vector),t,cubic):
```

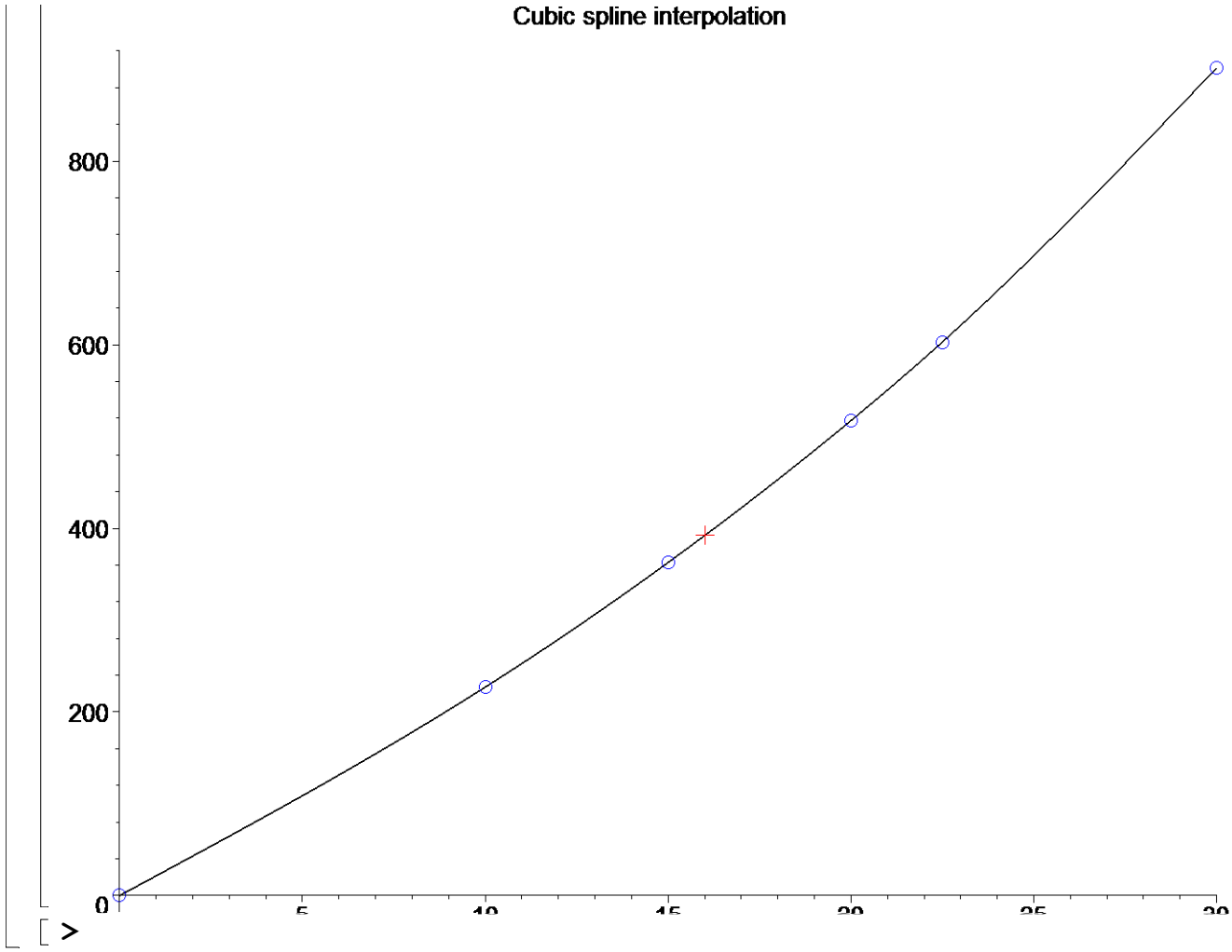
Computing the value of the function at the desired value of X,

```
> fcubic(xdesired);
```

392.1542016

Plotting the cubic spline interpolant and the value of Y at the desired calue of X,

```
> plot([xy,[[xdesired,fcubic(xdesired)]],fcubic],ranger(x,n),style=[POINT,POINT,LINE],color=[BLUE,RED,BLACK],symbol=[CIRCLE,CROSS],symbolsize=[30,40],thickness=2,title="Cubic spline interpolation");
```



Section VI: Conclusion.

Maple helped us to apply our knowledge of numerical methods of interpolation to find the value of y at a particular value of x using linear and quadratic spline interpolation. Using Maple functions and plotting routines made it easy to illustrate this method.

References

[1] *Nathan Collier, Autar Kaw, Jai Paul , Michael Keteltas, Holistic Numerical Methods Institute, See http://numericalmethods.eng.usf.edu/mws/gen/05inp/mws_gen_inp_sim_spline.mws http://numericalmethods.eng.usf.edu/mws/gen/05inp/mws_gen_inp_txt_spline.pdf*