Newton Divided Difference Method of Interpolation--Graphical

© 2003 Nathan Collier, Autar Kaw, Jai Paul , Michael Keteltas, University of South Florida ,
kaw@eng.usf.edu , http://numericalmethods.eng.usf.edu/mws

NOTE: This worksheet demonstrates the use of Maple to illustrate the Newton's Divided Difference Method of interpolation. We limit this worksheet to using first, second, and third order polynomials.

Introduction

The Newton's Divided Difference Polynomial method of interpolation (for detailed explanation, you can read the textbook notes and examples, and see a Power Point Presentation) is based on the following.

The general form of the Newton's divided difference polynomial for $(n+1)$ data points $(x_0,y_0),(x_1,y_1),.....,(x_{n-1},y_{n-1}),(x_n,y_n)$ is given as

$f_n(x)=b_0+b_1(x-x_0)+....+b_n(x-x_0)(x-x_1)...(x-x_{n-1})$ where n is the order of the polynomial that approximates the function and,

$$b_0=f[x_0]$$
$$b_1=f[x_1,x_0]$$
$$b_2=f[x_2,x_1,x_0]$$
$$.....$$
$$b_{n-1}=f[x_{n-1},x_{n-2},....,x_0]$$
$$b_n=f[x_n,x_{n-1},....,x_0]$$

where the definition of m^{th} divided difference is $b_m=f[x_m,.....,x_0]$

$$=\frac{f[x_m,.....,x_1]-f[x_{m-1},.....,x_0]}{x_m-x_0}$$

The examples that follow illustrate the method in a descriptive manner. For further explanations, click on text notes for the method on the website.

```
> 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 : Input 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.

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

The following function considers the x data and selects those data points which are close to the desired x value based on the least absolute difference between the x values and the desired x value. This function selects the two closest data points that bracket the desired value of x. It first picks the closest data point to the desired x value. It then checks if this value is less than or greater than the desired value.If it is less, then, it selects the data point which is greater than the desired value and also the closest, and viceversa.

Finds the absolute difference between the X values and the desired X value.

```
> for i from 1 to n do
    co[i]:=abs(x[i]-xdesired);
  end do:
```

Identifies the X value with the least absolute difference.

```
c:=co[1]:
for i from 2 to n do
  if c > co[i] then
    c:=co[i];
    ci:=i;
  end if:
end do:
```

If the value with the least absolute difference is less than the desired value, then it selects the closest data point greater than the desired value to bracket the desired value.

```
if x[ci] < xdesired then
  q:=1;
  for i from 1 to n do
    if x[i] > xdesired then
      nex[q]:=x[i];
      q:=q+1;
    end if;
```

```
  end do;

  b:=nex[1]:
  for i from 2 to q-1 do
    if b > nex[i] then
      b:=nex[i];
    end if:
  end do:

  for i from 1 to n do
    if x[i]=b then bi:=i end if;
  end do;
end if:
```

If the value with the least absolute difference is greater than the desired value, then it selects the closest data point less than the desired value to bracket the desired value.

```
if x[ci] > xdesired then
  q:=1;
  for i from 1 to n do
    if x[i] < xdesired then
      nex[q]:=x[i];
      q:=q+1;
    end if;
  end do;

  b:=nex[1]:
  for i from 2 to q-1 do
    if b < nex[i] then
      b:=nex[i];
    end if:
  end do:

  for i from 1 to n do
    if x[i]=b then bi:=i end if;
  end do;
end if:

firsttwo:=<ci,bi>:
```

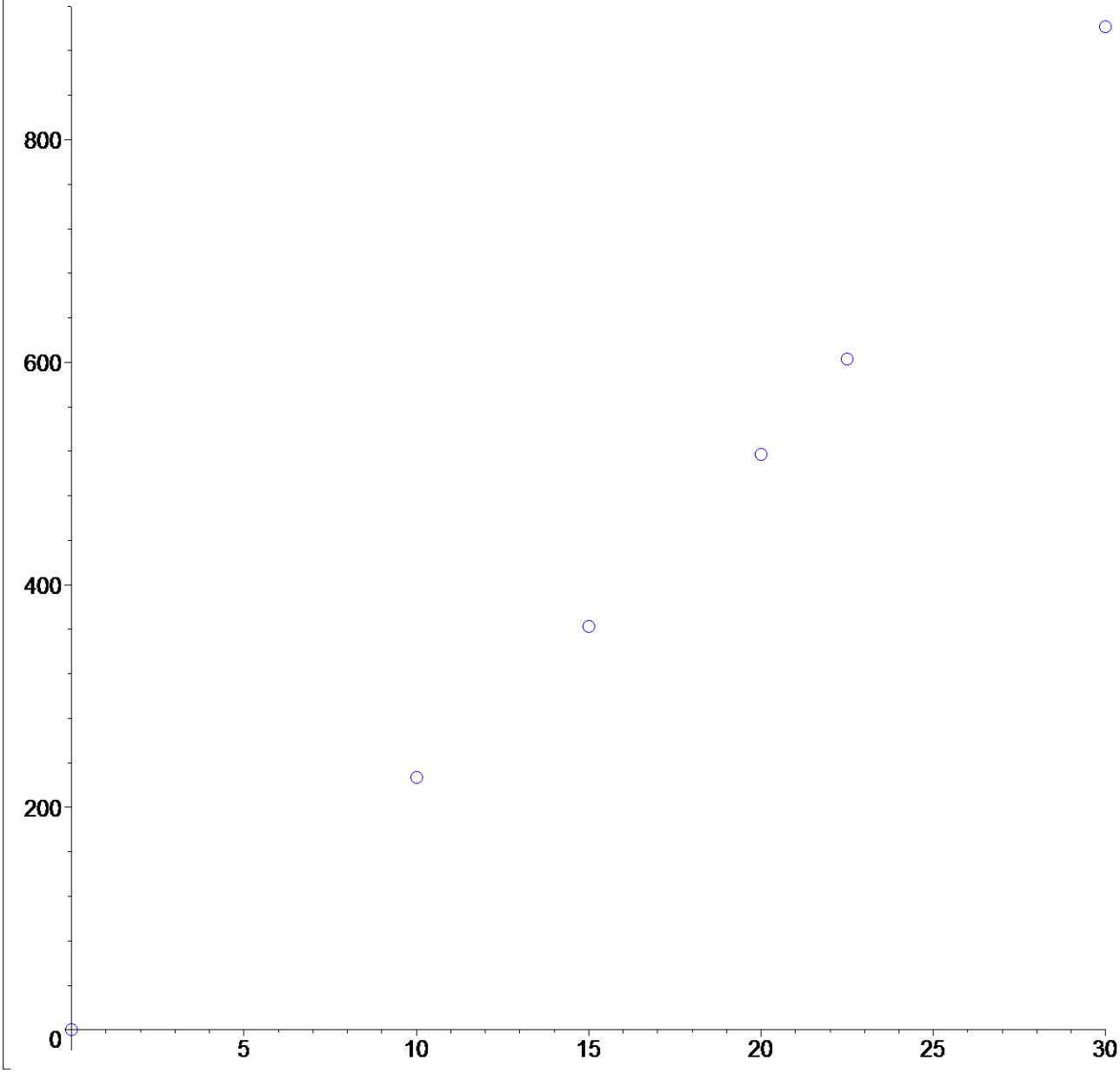
If more than two values are desired, the same procedure as above is followed to choose the 2 datapoints which bracket the desired value. In addition, the following function selects the subsequent values that are closest to the desired value and puts all the values into a matrix, maintaining the original data order.

```
> for i from 1 to n do
    A[i,2]:=i;
    A[i,1]:=co[i];
  end do:

  for passnum from 1 to n-1 do
    for i from 1 to n-passnum do
      if A[i,1]>A[i+1,1] then
        temp1:=A[i,1];
        temp2:=A[i,2];
        A[i,1]:=A[i+1,1];
        A[i,2]:=A[i+1,2];
        A[i+1,1]:=temp1;
        A[i+1,2]:=temp2;
      end if:
    end do:
  end do:

  for i from 1 to n do
    A[i,3]:=i;
  end do:

  for passnum from 1 to n-1 do
    for i from 1 to n-passnum do
      if A[i,2]>A[i+1,2] then
        temp1:=A[i,1];
        temp2:=A[i,2];
        temp3:=A[i,3];
        A[i,1]:=A[i+1,1];
        A[i,2]:=A[i+1,2];
        A[i,3]:=A[i+1,3];
        A[i+1,1]:=temp1;
        A[i+1,2]:=temp2;
        A[i+1,3]:=temp3;
      end if:
    end do:
  end do:

  for i from 1 to n do
    d[i]:=A[i,3];
  end do:
```

```
  if d[firsttwo[2]]<>2 then
    temp:=d[firsttwo[2]];
    d[firsttwo[2]]:=1;
    for i from 1 to n do
      if i <> firsttwo[1] and i <> firsttwo[2] and d[i] <= temp
  then
         d[i]:=d[i]+1;
      end if:
    end do;
    d[firsttwo[1]]:=1;
  end if:
> ranger:=proc(ar,n)
  local i,xmin,xmax,xrange;
  xmin:=ar[1]:
  xmax:=ar[1]:
  for i from 1 to n do
  if ar[i] > xmax then xmax:=ar[i] end if;
  if ar[i] < xmin then xmin:=ar[i] 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 Interpolation.

Given $(x_0, y_0), (x_1, y_1),$ fit a linear interpolant through the data. Noting $y_0 = f(x_0)$ and $y_1 = f(x_1)$, assume the linear interpolant, $f_1(x)$ is given by $f_1(x) = b_0 + b_1(x - x_0)$, where $b_0 = f(x_0)$ and $b_1 = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$.

Hence, the two closest data points to the desired value are chosen in this method.

```
> datapoints:=2:
> p:=1:
  for i from 1 to n do
```

```
  if d[i] <= datapoints then
  xdata[p]:=x[i];
  ydata[p]:=y[i];
  p:=p+1;
  end if
  end do:
```

```
> entries(xdata);
```

$$[20], [15]$$

```
> entries(ydata);
```

$$[517.35], [362.78]$$

Calculating coefficients of Newton's Divided difference polynomial

```
> b0:=ydata[1];
```

$$b0 := 517.35$$

```
> b1:=(ydata[2]-ydata[1])/(xdata[2]-xdata[1]);
```

$$b1 := 30.91400000$$

Newton's divided difference formula for linear interpolation is

```
> flinear(z):=b0+b1*(z-xdata[1]);
```

$$\text{flinear}(z) := -100.9300000 + 30.91400000\, z$$

Substituting the value of the desired X value for z in the above equation, the corresponding Y value is found.

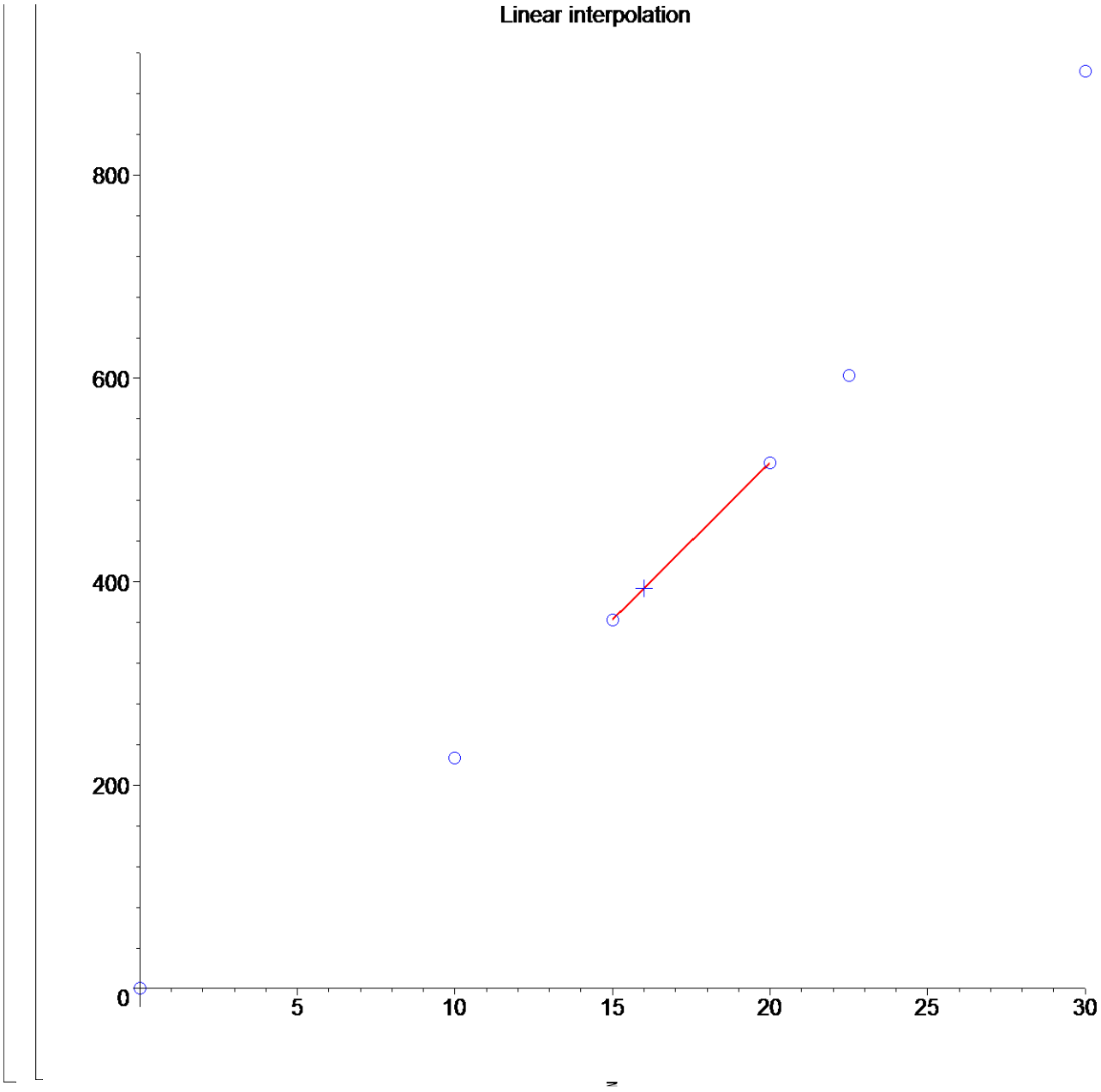
```
> eval(flinear(z),z=xdesired);
```

$$393.6940000$$

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

Plotting the Linear interpolant and the value of Y for the desired X

```
> plot([[t,eval(flinear(z),z=t),t=ranger(xdata,datapoints)],xy,[[
  xdesired,eval(flinear(z),z=xdesired)]]],z=ranger(x,n),style=[LI
  NE,POINT,POINT],color=[RED,BLUE,BLUE],symbol=[CROSS,CIRCLE],sym
  bolsize=[40,30],thickness=3,title="Linear interpolation");
```



Section IV: Quadratic Interpolation (Second order polynomial)

Given (x_0, y_0), (x_1, y_1), and (x_2, y_2), fit a quadratic interpolant through the data. Noting $y = f(x)$, $y_0 = f(x_0)$, $y_1 = f(x_1)$, and $y_2 = f(x_2)$, assume the quadratic interpolant $f_2(x)$ given by $f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$, where

$$b_0 = f(x_0)$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}.$$

Hence, the three closest data points to the desired value are chosen in this method.

```
> datapoints:=3:
> p:=1:
  for i from 1 to n do
  if d[i] <= datapoints then
  xdata[p]:=x[i];
  ydata[p]:=y[i];
  p:=p+1;
  end if
  end do:
```

```
> entries(xdata);
```

$$[10], [20], [15]$$

```
> entries(ydata);
```

$$[227.04], [517.35], [362.78]$$

Calculating coefficients of Newton's Divided difference polynomial

```
> b0:=ydata[1];
```

$$b0 := 227.04$$

```
> b1:=(ydata[2]-ydata[1])/(xdata[2]-xdata[1]);
```

$$b1 := 29.03100000$$

```
> b2:=((ydata[3]-ydata[2])/(xdata[3]-xdata[2])-(ydata[2]-ydata[1]
  )/(xdata[2]-xdata[1]))/(xdata[3]-xdata[1]);
```

$$b2 := 0.3766000000$$

Newton's divided difference formula for quadratic interpolation is

```
> fquad(z):=b0+b1*(z-xdata[1])+b2*(z-xdata[1])*(z-xdata[2]);
```

$$\text{fquad}(z) := -63.2700000 + 29.03100000\, z + 0.3766000000\, (z - 10)\, (z - 20)$$

Substituting the value of the desired X value for z in the above equation, the corresponding Y value is found.

```
> eval(fquad(z),z=xdesired);
```

$$392.1876000$$

```
> fnew:=%:
>
```

The absolute percentage relative approximate error between the first order and second order interpolation is

```
> epsilon:=abs((fnew-fprev)/fnew)*100;
```

$$\varepsilon := 0.3841018941$$

The number of significant digits at least correct in the solution is

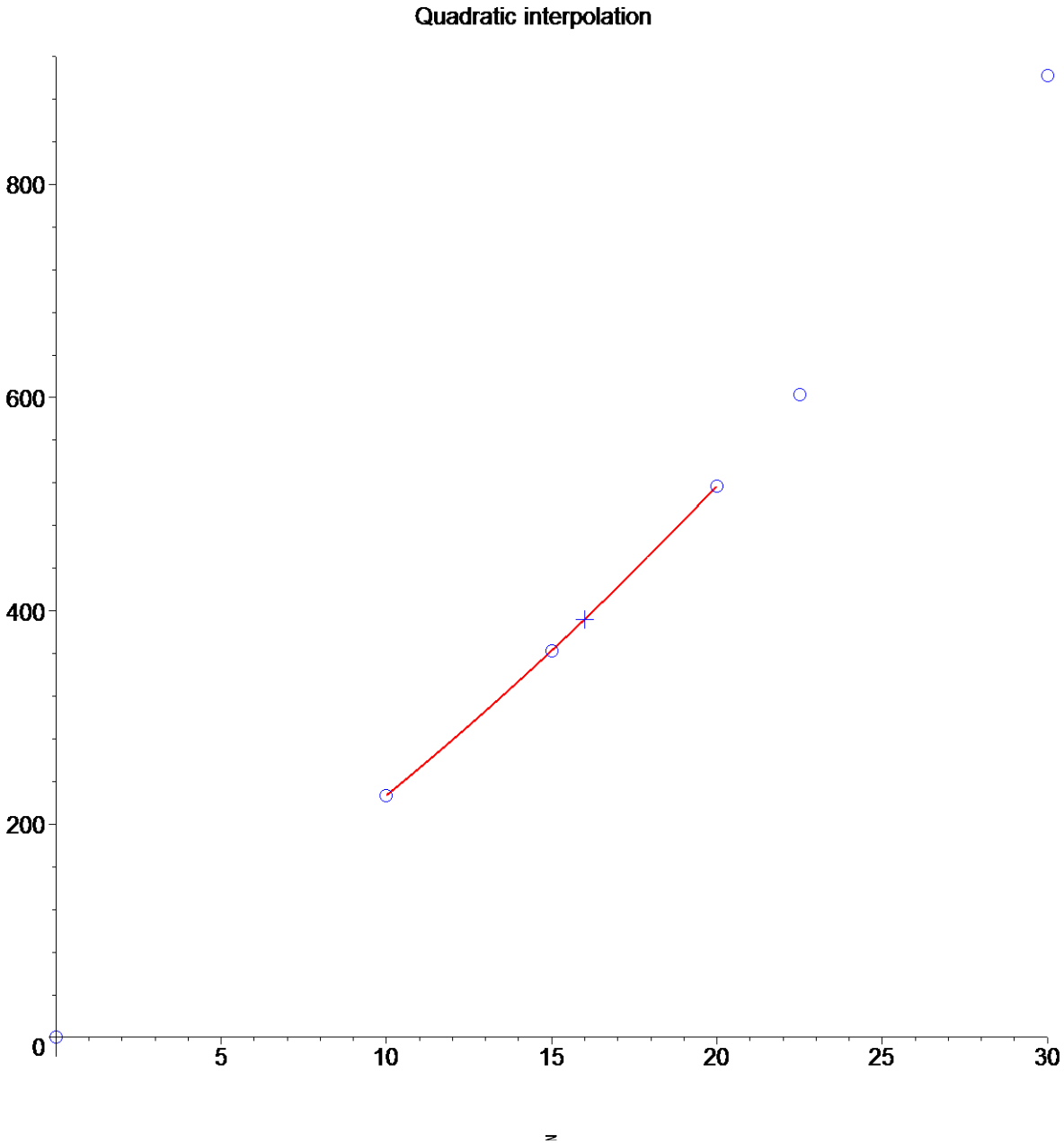
```
> sigdig:=floor(2-log10(epsilon/0.5));
```

$$sigdig := 2$$

```
> fprev:=fnew:
```

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

```
> plot([[t,eval(fquad(z),z=t),t=ranger(xdata,datapoints)],xy,[[xd
  esired,eval(fquad(z),z=xdesired)]]],z=ranger(x,n),style=[LINE,P
  OINT,POINT],color=[RED,BLUE,BLUE],symbol=[CROSS,CIRCLE],symbols
  ize=[40,30],thickness=3,title="Quadratic interpolation");
```



Section V: Cubic Interpolation (Third order polynomial)

The same method as above is used for 4 data points and the corresponding coefficients, are calcualted and evaluated in the formula as shown below to find the corresponding Y value for the

desired X value.
Hence, the four closest data points to the desired value are chosen in this method.

```
> datapoints:=4:
> p:=1:
  for i from 1 to n do
  if d[i] <= datapoints then
  xdata[p]:=x[i];
  ydata[p]:=y[i];
  p:=p+1;
  end if
  end do:
```

```
> entries(xdata);
```

$$[10], [20], [15], [22.5]$$

```
> entries(ydata);
```

$$[227.04], [517.35], [362.78], [602.97]$$

Calculating coefficients of Newton's Divided difference polynomial

```
> b0:=ydata[1];
```

$$b0 := 227.04$$

```
> b1:=(ydata[2]-ydata[1])/(xdata[2]-xdata[1]);
```

$$b1 := 29.03100000$$

```
> b2:=((ydata[3]-ydata[2])/(xdata[3]-xdata[2])-(ydata[2]-ydata[1]
  )/(xdata[2]-xdata[1]))/(xdata[3]-xdata[1]);
```

$$b2 := 0.3766000000$$

```
> b3:=(((ydata[4]-ydata[3])/(xdata[4]-xdata[3])-(ydata[3]-ydata[2
  ])/(xdata[3]-xdata[2]))/(xdata[4]-xdata[2])-b2)/(xdata[4]-xdata
  [1]);
```

$$b3 := 0.005434666560$$

Newton's divided difference formula for cubic interpolation is

```
> fcubic(z):=b0+b1*(z-xdata[1])+b2*(z-xdata[1])*(z-xdata[2])+b3*(
  z-xdata[1])*(z-xdata[2])*(z-xdata[3]);
```

$$\text{fcubic}(z) := -63.2700000 + 29.03100000\, z + 0.3766000000\, (z - 10)\, (z - 20) + 0.005434666560\, (z - 10)\, (z - 20)\, (z - 15)$$

Substituting the value of the desired X value for z in the above equation, the corresponding Y value is found.

```
> eval(fcubic(z),z=xdesired);
```

$$392.0571680$$

```
> fnew:=%:
```

The absolute percentage relative approximate error between the second order and third order interpolation is

```
> epsilon:=abs((fnew-fprev)/fnew)*100;
```

$$\varepsilon := 0.03326861760$$

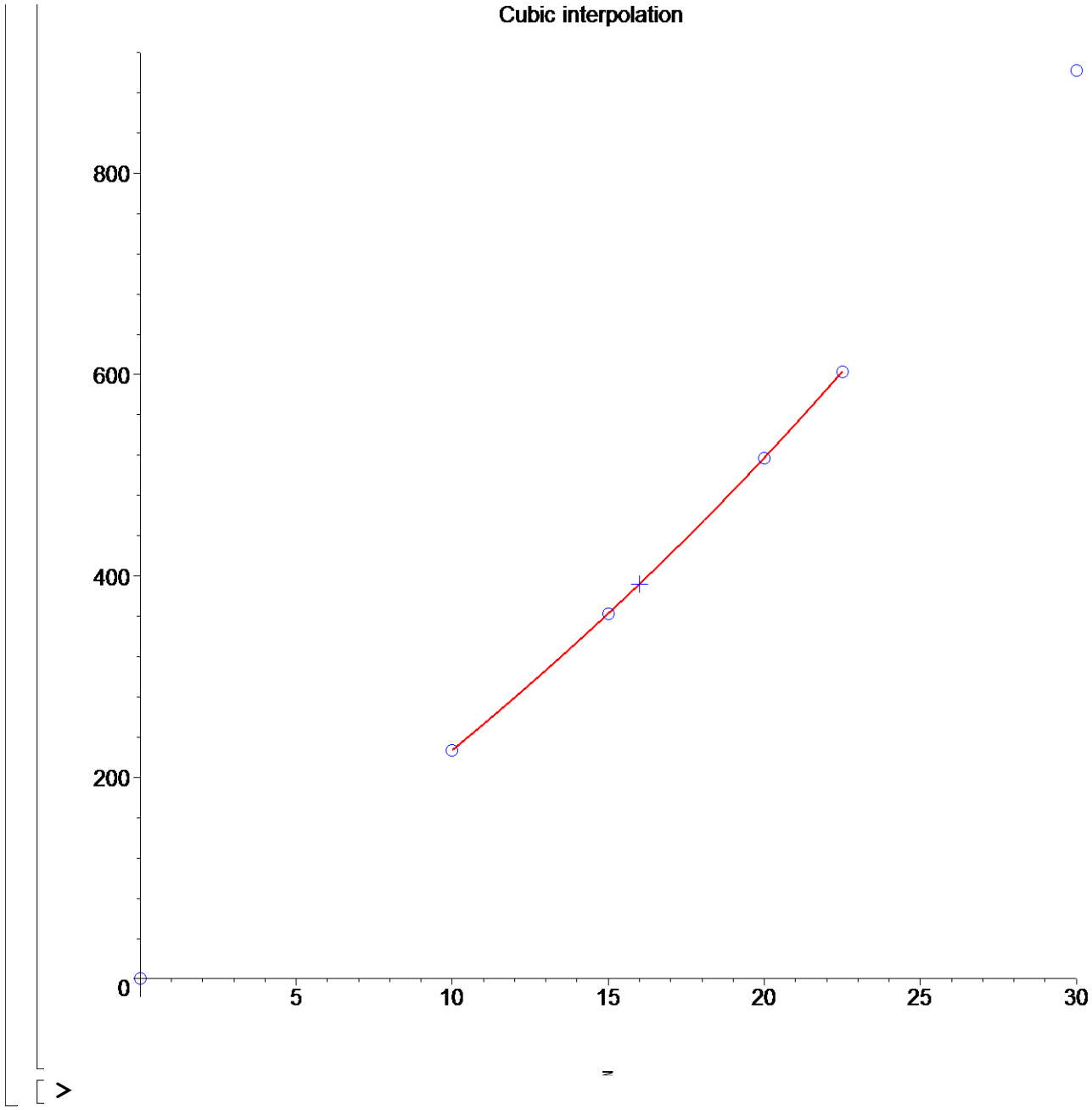
The number of significant digits at least correct in the solution

```
> sigdig:=floor(2-log10(epsilon/0.5));
```

$$sigdig := 3$$

Plotting the cubic interpolant and the value of Y for the desired X

```
> plot([[t,eval(fcubic(z),z=t),t=ranger(xdata,datapoints)],xy,[[x
  desired,eval(fcubic(z),z=xdesired)]]],z=ranger(x,n),style=[LINE
  ,POINT,POINT],color=[RED,BLUE,BLUE],symbol=[CROSS,CIRCLE],symbo
  lsize=[40,30],thickness=3,title="Cubic 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 first, second, and third order Newton's Divided Difference Polynomial method of 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_ndd.mws
http://numericalmethods.eng.usf.edu/mws/gen/05inp/mws_gen_inp_txt_ndd.pdf

Disclaimer*:* While every effort has been made to validate the solutions in this worksheet, University of South Florida and the contributors are not responsible for any errors contained and are not liable for any damages resulting from the use of this material.