Topic : Lagrangian Method
Simulation : Graphical Simulation of the Method
Language : Mathmatica 4.1
Authors : Nathan Collier, Autar Kaw
Date : 15 July 2002
Abstract : This simulation illustrates the Lagrangian method of interpolation. Given n data points of y versus x, you are then required to find the value of y at a particular value of x using first, second, and third order interpolation. So on has to first pick the needed data point, and then use those points to interpolate the data

■ INPUTS: Enter the Following

Array of x data

```
In[965]:= x = {10, 0, 20, 15, 30, 22.5};
```

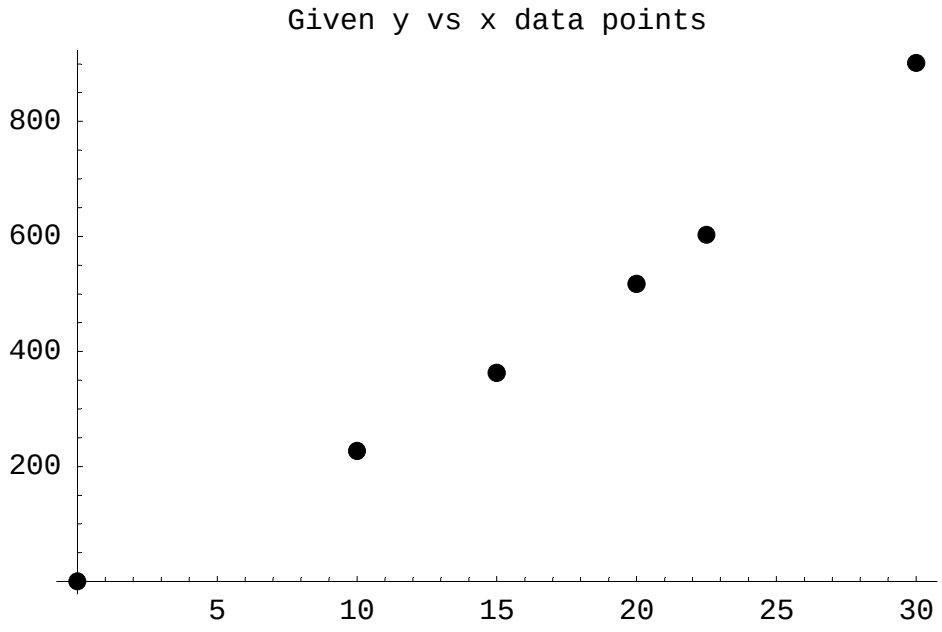
Array of y data

```
In[966]:= y = {227.04, 0, 517.35, 362.78, 901.67, 602.97};
```

Value of x at which y is desired

```
In[967]:= xdesired := 16

In[968]:= nn := Abs[Dimensions[x]]

In[969]:= n := nn[[1]]

In[970]:= xy = Table[0, {i, n}, {j, 2}];

In[971]:= Do[xy[[i, 1]] = x[[i]]; xy[[i, 2]] = y[[i]], {i, 1, n}]
```

In[972]:=
```
data = ListPlot[xy, PlotStyle → PointSize[0.02],
   PlotLabel -> "Given y vs x data points", TextStyle → {FontSize → 11}];
```



■ SOLUTION

The following considers the x and y data and selects the two closest data points that bracket the desired value of x.

In[973]:=
```
comp := Abs[x - xdesired]
```

In[974]:=
```
c := Min[comp]
```

In[975]:=
```
Do[If[comp[[i]] == c, ci = i], {i, 1, n}]
```

In[976]:=
```
R = Table[0, {i, 1, n}];
```

In[977]:=
```
If[x[[ci]] < xdesired, q = 1;
 Do[If[x[[i]] > xdesired, R[[q]] = x[[i]]; q = q + 1], {i, 1, n}];
 RR = Table[RR[[i]] = R[[i]], {i, 1, q - 1}];
 b = Min[RR]; Do[If[x[[i]] == b, bi = i], {i, 1, n}]]
```

In[978]:=
```
If[x[[ci]] > xdesired, q = 1;
 Do[If[x[[i]] < xdesired, R[[q]] = x[[i]]; q = q + 1], {i, 1, n}];
 RR = Table[RR[[i]] = R[[i]], {i, 1, q - 1}];
 b = Max[RR]; Do[If[x[[i]] == b, bi = i], {i, 1, n}]]
```

In[979]:=
```
firsttwo := {ci, bi}
```

If more than two values are desired, the following selects the subsequent values and puts all the values into a matrix, maintaining the original data order.

In[980]:=
```
A = Table[0, {i, n}, {j, 3}];
```

In[981]:=
```
Do[A[[i, 2]] = i; A[[i, 1]] = comp[[i]], {i, 1, n}]
```

In[982]:=
```
A = Sort[A];
```

In[983]:=
```
Do[A[[i, 3]] = i, {i, 1, n}]
```

In[984]:=
```
T = A[[All, 1]];
```

In[985]:=
```
Do[A[[i, 1]] = A[[i, 2]]; A[[i, 2]] = T[[i]], {i, 1, n}]
```

In[986]:=
```
A = Sort[A];
```

In[987]:=
```
d = A[[All, 3]];
```

In[988]:=
```
If[d[[firsttwo[[2]]]] ≠ 2, temp = d[[firsttwo[[2]]]]; d[[firsttwo[[2]]]] = 1;
 Do[If[i ≠ firsttwo[[2]] && i ≠ firsttwo[[1]] && d[[i]] ≤ temp,
   d[[i]] = d[[i]] + 1]; d[[firsttwo[[1]]]] = 1, {i, 1, n}]]
```

Linear Interpolation

Pick two data points

In[989]:=
```
datapoints = 2;
```

In[990]:=
```
xData = Table[0, {i, 1, datapoints}];
yData = Table[0, {i, 1, datapoints}];
```

In[992]:=
```
p = 1; Do[If[d[[i]] ≤ datapoints,
  xData[[p]] = x[[i]]; yData[[p]] = y[[i]]; p = p + 1], {i, 1, n}]
```

In[993]:=
```
xData // MatrixForm
```

Out[993]//MatrixForm=

$$\begin{pmatrix} 20 \\ 15 \end{pmatrix}$$

In[994]:=
```
yData // MatrixForm
```

Out[994]//MatrixForm=

$$\begin{pmatrix} 517.35 \\ 362.78 \end{pmatrix}$$

In[995]:=
```
L0[x_] := (x - xData[[2]]) / (xData[[1]] - xData[[2]])
```

In[996]:=
```
L0[xdesired]
```

Out[996]= $\frac{1}{5}$

In[997]:=
```
L1[x_] := (x - xData[[1]]) / (xData[[2]] - xData[[1]])
```

In[998]:=
```
L1[xdesired]
```

Out[998]= $\frac{4}{5}$

In[999]:=
```
L0[xdesired] + L1[xdesired]
```

Out[999]= 1

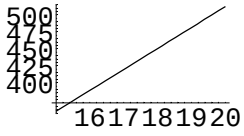
In[1000]:=
```
f[x_] := L0[x] * yData[[1]] + L1[x] * yData[[2]]
```

In[1001]:=
```
f[xdesired]
```

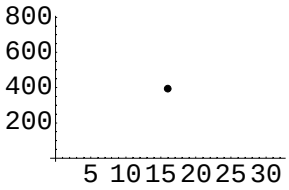
Out[1001]= 393.694

In[1002]:=
```
fprev = %;
```

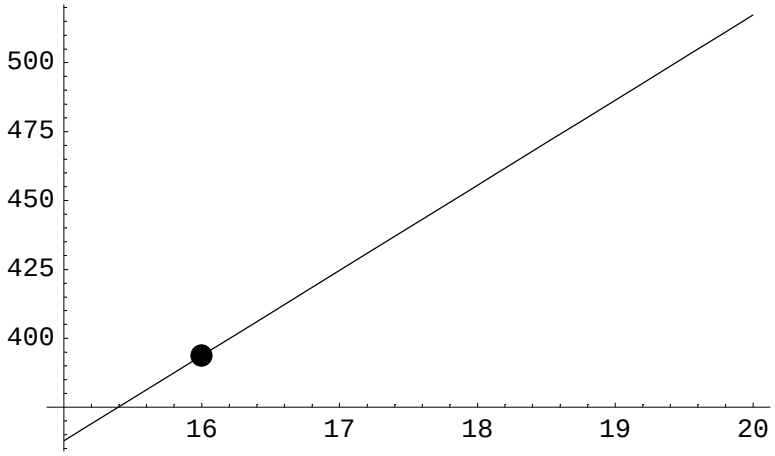
In[1003]:=
```
lin = Plot[f[x], {x, Min[xData], Max[xData]}];
```



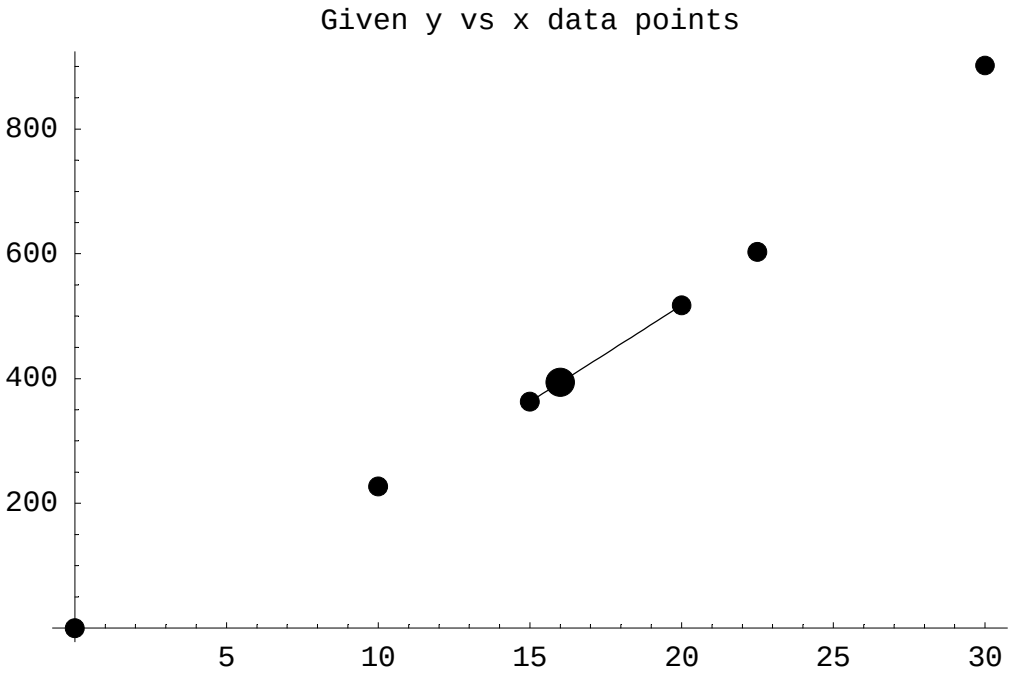
In[1004]:=
```
desire = ListPlot[{{xdesired, f[xdesired]}}, PlotStyle → PointSize[0.03]];
```



In[1005]:= **Show[desire, lin];**



In[1006]:= **Show[data, lin, desire, PlotRange → All];**



Quadratic Interpolation

Pick two data points

In[1007]:= **datapoints = 3;**

In[1008]:=
```
xData = Table[0, {i, 1, datapoints}];
yData = Table[0, {i, 1, datapoints}];
```

In[1010]:=
```
p = 1; Do[If[d[[i]] ≤ datapoints,
  xData[[p]] = x[[i]]; yData[[p]] = y[[i]]; p = p + 1], {i, 1, n}]
```

In[1011]:=
```
xData // MatrixForm
```

Out[1011]//MatrixForm=

$$\begin{pmatrix} 10 \\ 20 \\ 15 \end{pmatrix}$$

In[1012]:=
```
yData // MatrixForm
```

Out[1012]//MatrixForm=

$$\begin{pmatrix} 227.04 \\ 517.35 \\ 362.78 \end{pmatrix}$$

In[1013]:=
```
L0[x_] := ((x - xData[[2]]) * (x - xData[[3]])) /
  ((xData[[1]] - xData[[2]]) * (xData[[1]] - xData[[3]]))
```

In[1014]:=
```
L0[xdesired]
```

Out[1014]= $-\frac{2}{25}$

In[1015]:=
```
L1[x_] := ((x - xData[[1]]) * (x - xData[[3]])) /
  ((xData[[2]] - xData[[1]]) * (xData[[2]] - xData[[3]]))
```

In[1016]:=
```
L1[xdesired]
```

Out[1016]= $\frac{3}{25}$

In[1017]:=
```
L2[x_] := ((x - xData[[1]]) * (x - xData[[2]])) /
  ((xData[[3]] - xData[[1]]) * (xData[[3]] - xData[[2]]))
```

In[1018]:=
```
L2[xdesired]
```

Out[1018]= $\frac{24}{25}$

In[1019]:=
```
L0[xdesired] + L1[xdesired] + L2[xdesired]
```

Out[1019]= 1

In[1020]:=
```
f[x_] := L0[x] * yData[[1]] + L1[x] * yData[[2]] + L2[x] * yData[[3]]
```

In[1021]:=
```
f[xdesired]
```

Out[1021]= 392.188

In[1022]:=
```
fnew = %;
```

In[1023]:=
```
ϵa = Abs[(fnew - fprev) / fnew * 100]
```
Out[1023]=
```
0.384102
```

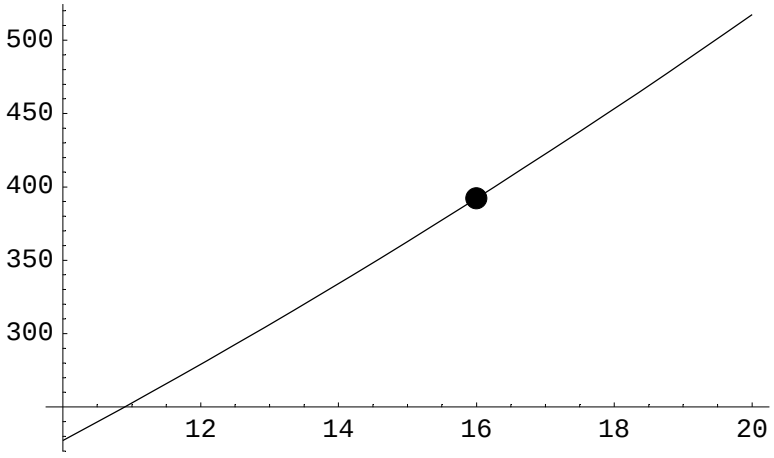
In[1024]:=
```
sigdig = Floor[2 - Log[10, (ϵa / 0.5)]]
```
Out[1024]=
```
2
```

In[1025]:=
```
fprev = %%%;
```

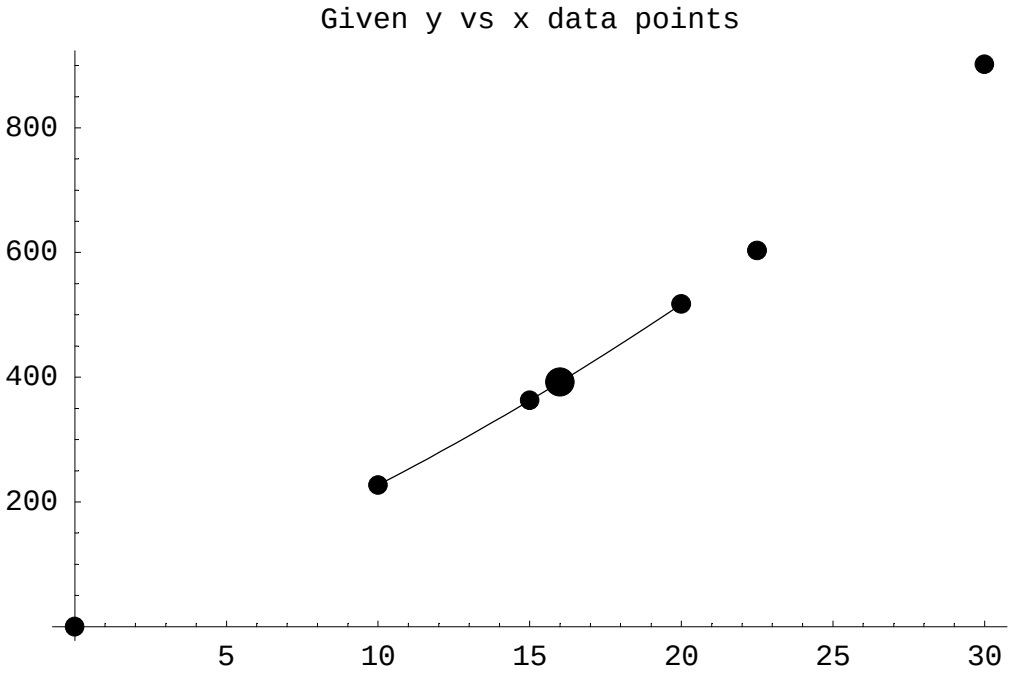
In[1026]:=
```
lin = Plot[f[x], {x, Min[xData], Max[xData]}];
```

In[1027]:=
```
desire = ListPlot[{{xdesired, f[xdesired]}}, PlotStyle → PointSize[0.03]];
```

In[1028]:=
```
Show[desire, lin];
```



In[1029]:=
```
Show[data, lin, desire, PlotRange → All];
```



Cubic Interpolation

Pick four data points

In[1030]:=
```
datapoints = 4;
```

In[1031]:=
```
xData = Table[0, {i, 1, datapoints}];
yData = Table[0, {i, 1, datapoints}];
```

In[1033]:=
```
p = 1; Do[If[d[[i]] ≤ datapoints,
  xData[[p]] = x[[i]]; yData[[p]] = y[[i]]; p = p + 1], {i, 1, n}]
```

In[1034]:=
```
xData // MatrixForm
```

Out[1034]//MatrixForm=

$$\begin{pmatrix} 10 \\ 20 \\ 15 \\ 22.5 \end{pmatrix}$$

In[1035]:=
```
yData // MatrixForm
```

Out[1035]//MatrixForm=

$$\begin{pmatrix} 227.04 \\ 517.35 \\ 362.78 \\ 602.97 \end{pmatrix}$$

In[1036]:=
```
L0[x_] := ((x - xData[[2]]) * (x - xData[[3]]) * (x - xData[[4]])) /
  ((xData[[1]] - xData[[2]]) *
    (xData[[1]] - xData[[3]]) * (xData[[1]] - xData[[4]]))
```

In[1037]:=
```
L0[xdesired]
```

Out[1037]= -0.0416

In[1038]:=
```
L1[x_] := ((x - xData[[1]]) * (x - xData[[3]]) * (x - xData[[4]])) /
  ((xData[[2]] - xData[[1]]) *
    (xData[[2]] - xData[[3]]) * (xData[[2]] - xData[[4]]))
```

In[1039]:=
```
L1[xdesired]
```

Out[1039]= 0.312

In[1040]:=
```
L2[x_] := ((x - xData[[1]]) * (x - xData[[2]]) * (x - xData[[4]])) /
  ((xData[[3]] - xData[[1]]) *
    (xData[[3]] - xData[[2]]) * (xData[[3]] - xData[[4]]))
```

In[1041]:=
```
L2[xdesired]
```

Out[1041]= 0.832

In[1042]:=
```
L3[x_] := ((x - xData[[1]]) * (x - xData[[2]]) * (x - xData[[3]])) /
  ((xData[[4]] - xData[[1]]) *
    (xData[[4]] - xData[[2]]) * (xData[[4]] - xData[[3]]))
```

In[1043]:=
```
L3[xdesired]
```

Out[1043]= -0.1024

In[1044]:=
```
L0[xdesired] + L1[xdesired] + L2[xdesired] + L3[xdesired]
```

Out[1044]= 1.

In[1045]:=
```
f[x_] :=
 L0[x] * yData[[1]] + L1[x] * yData[[2]] + L2[x] * yData[[3]] + L3[x] * yData[[4]]
```

In[1046]:=
```
f[xdesired]
```

Out[1046]= 392.057

In[1047]:=
```
fnew = %;
```

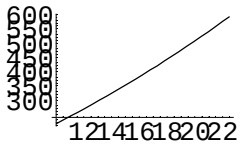
In[1048]:=
```
∈a = Abs[(fnew - fprev) / fnew * 100]
```

Out[1048]=
```
0.0332686
```

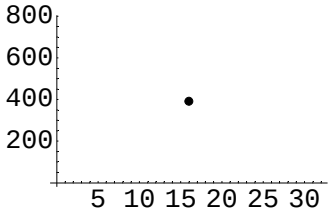
In[1049]:=
```
sigdig = Floor[2 - Log[10, (∈a / 0.5)]]
```

Out[1049]=
```
3
```

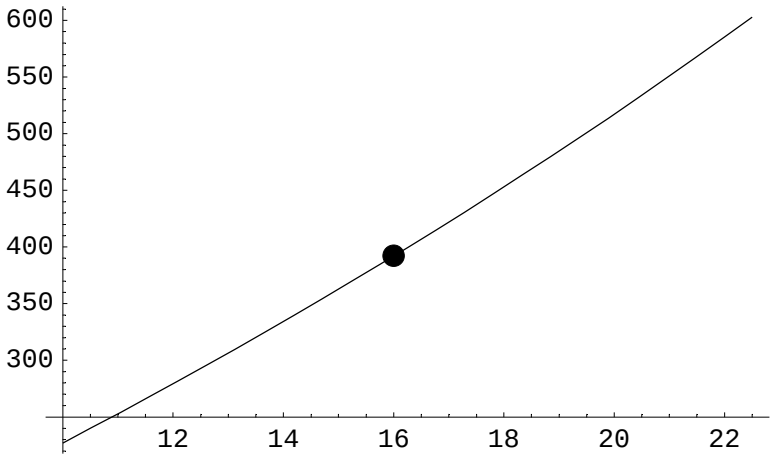
In[1050]:=
```
lin = Plot[f[x], {x, Min[xData], Max[xData]}];
```



In[1051]:=
```
desire = ListPlot[{{xdesired, f[xdesired]}}, PlotStyle → PointSize[0.03]];
```



In[1052]:=
```
Show[desire, lin];
```



In[1053]:= **Show[data, lin, desire, PlotRange → All];**