Mat360 Homework

Tea Time Numerical Analysis
Section 3.2: Lagrange Polynomials

Andy Long, Spring, 2024

pp. 121-122, #1, 2 (find two polynomials!), 8, 9, 10, 13, 22

Preliminaries:
Lagrange Polynomials make fitting points easy!

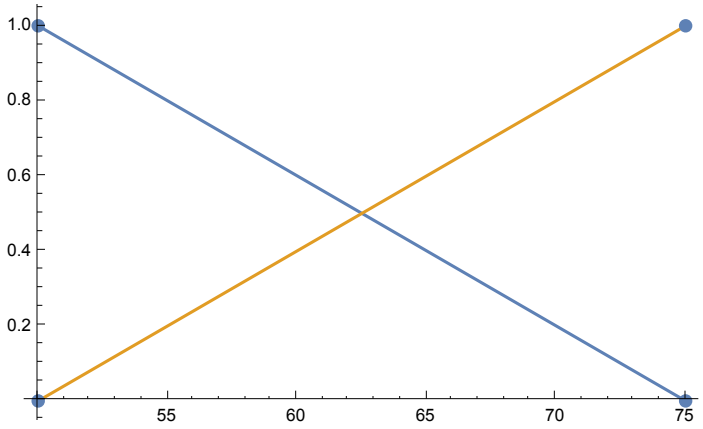
Lagrange's linear functions are linear functions which are zero at one value of x, and one at the other:

In[379]:=
```
lagrange[x_, x1_, x2_] := (x - x2) / (x1 - x2) (* 1 at x1, 0 at x2 *)
{x0, x1} = {50, 75}
{y0, y1} = {180, 230}
Show[
 Plot[{lagrange[x, x0, x1], lagrange[x, x1, x0]}, {x, x0, x1}],
 ListPlot[
  {{x0, 0}, {x1, 0}, {x0, 1}, {x1, 1}},
  PlotStyle → {PointSize → Large}]
]
```

Out[380]= {50, 75}

Out[381]= {180, 230}

Out[382]=

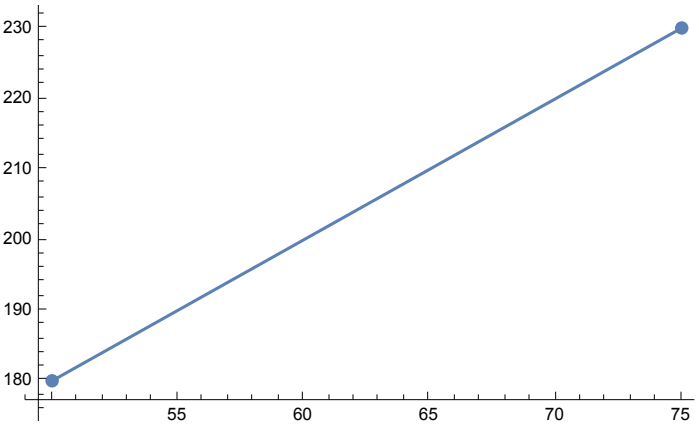


Lagrange linear polynomials are just sums of two Lagrange linear functions that fit two points

Let's use these Lagrange linear functions fit two points: (50,180), (75, 230):

In[383]:=
```
L1[x_, x0_, x1_, y0_, y1_] := y0 lagrange[x, x0, x1] + y1 lagrange[x, x1, x0]
p1 = ListPlot[{{x0, y0}, {x1, y1}}, PlotStyle → {PointSize → Large}];
p2 = Plot[L1[x, x0, x1, y0, y1], {x, x0, x1}];
Show[p1, p2]
```

Out[386]=



Exercise #1:

1. Write down the Lagrange interpolating polynomial passing through $(1, 2)$, $(1.5, -0.83)$, and $(2.11, -1)$.
2. Find a polynomial that passes through the four points

$$(0, 0),\ (1, 2),\ (4, -3),\ \text{and}\ (10, -1).$$

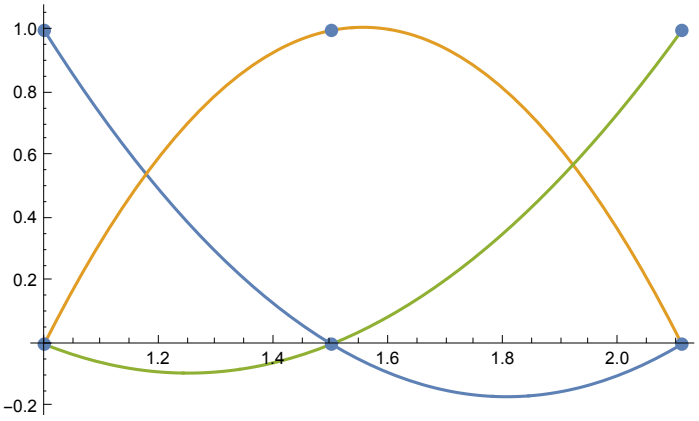
Lagrange quadratic functions are just products of Lagrange linear functions -- and they fit three points: two with function values zeros, and a function value of 1:

In[387]:=
```
lagrange2[x_, x1_, x2_, x3_] := lagrange[x, x1, x2] * lagrange[x, x1, x3]
(* lagrange2 is 1 at x1, zero at x2 and x3 *)
(* (1,2),(1.5,-0.83),and (2.11,-1) For problem #1: *)
{x0, x1, x2} = {1, 1.5, 2.11}
{y0, y1, y2} = {2, -0.83, -1}
Show[
 Plot[{lagrange2[x, x0, x1, x2],
   lagrange2[x, x1, x0, x2], lagrange2[x, x2, x0, x1]}, {x, x0, x2}],
 ListPlot[
  {{x0, 0}, {x1, 0}, {x2, 0}, {x0, 1}, {x1, 1}, {x2, 1}},
  PlotStyle → {PointSize → Large}]
]
```

Out[388]= {1, 1.5, 2.11}

Out[389]= {2, -0.83, -1}

Out[390]=



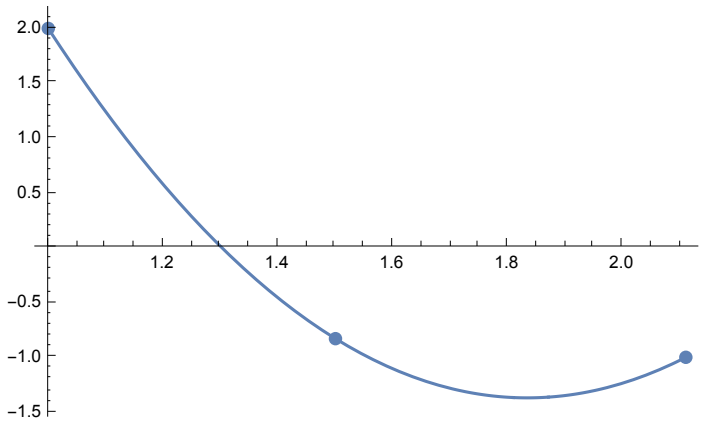
Lagrange quadratic polynomials are just sums of three Lagrange quadratic functions that each are zero at two points, and 1 at the other

Write down the Lagrange interpolating polynomial passing through (1, 2), (1.5, –0.83), and (2.11, –1).

In[391]:=
```
L2[x_, x0_, x1_, x2_, y0_, y1_, y2_] := y0 lagrange2[x, x0, x1, x2] +
  y1 lagrange2[x, x1, x0, x2] +
  y2 lagrange2[x, x2, x0, x1]
L2[x, x0, x1, x2]
p1 = ListPlot[{{x0, y0}, {x1, y1}, {x2, y2}}, PlotStyle → {PointSize → Large}];
p2 = Plot[L2[x, x0, x1, x2, y0, y1, y2], {x, x0, x2}];
Show[p2, p1]
```

Out[392]=
```
L2[x, 1, 1.5, 2.11]
```

Out[395]=



Exercise #2:

1. Write down the Lagrange interpolating polynomial passing through $(1, 2)$, $(1.5, -0.83)$, and $(2.11, -1)$.
2. Find a polynomial that passes through the four points

$$(0, 0),\ (1, 2),\ (4, -3),\ \text{and}\ (10, -1).$$

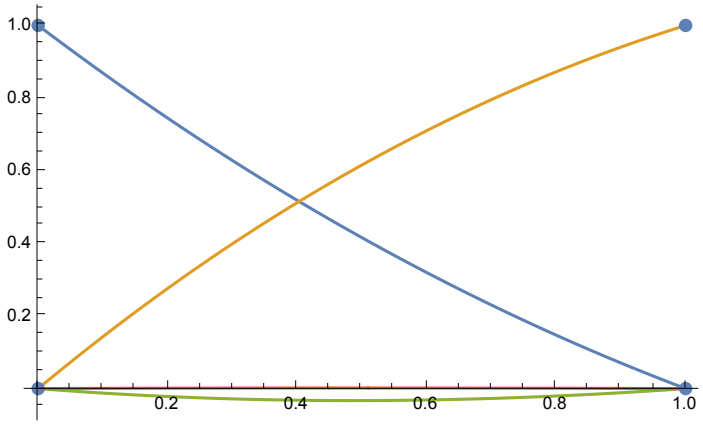
Lagrange cubic functions are also just products of Lagrange linear functions -- and they fit four points: three zeros and a 1 at the fourth point.

In[396]:=
```
lagrange3[x_, x1_, x2_, x3_, x4_] :=
 lagrange[x, x1, x2] * lagrange[x, x1, x3] * lagrange[x, x1, x4]
(* (0,0),(1,2),(4,-3),and (10,-1) *)
{x0, x1, x2, x3} = {0, 1, 4, 10}
{y0, y1, y2, y3} = {0, 2, -3, -1}
Show[
 Plot[{lagrange3[x, x0, x1, x2, x3], lagrange3[x, x1, x0, x2, x3],
   lagrange3[x, x2, x0, x1, x3], lagrange3[x, x3, x0, x1, x2]}, {x, x0, x1}],
 ListPlot[
  {{x0, 0}, {x1, 0}, {x2, 0}, {x3, 0}, {x0, 1}, {x1, 1}, {x2, 1}, {x3, 1}},
  PlotStyle → {PointSize → Large}]
]
```

Out[397]= {0, 1, 4, 10}

Out[398]= {0, 2, −3, −1}

Out[399]=



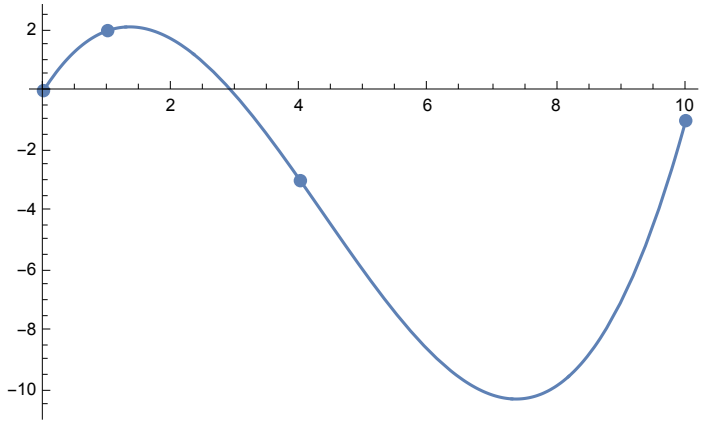
Lagrange cubic polynomials are just sums of four Lagrange cubic functions that each are zero at three points, and 1 at the other.

Let's fit four points of Exercise 2: (0, 0), (1, 2), (4, –3), and (10, –1)

In[400]:=
```
L3[x_, x0_, x1_, x2_, x3_, y0_, y1_, y2_, y3_] :=
 y0 lagrange3[x, x0, x1, x2, x3] + y1 lagrange3[x, x1, x2, x0, x3] +
  y2 lagrange3[x, x2, x0, x1, x3] + y3 lagrange3[x, x3, x2, x0, x1]
L3[x, x0, x1, x2, x3, y0, y1, y2, y3]
p1 =
  ListPlot[{{x0, y0}, {x1, y1}, {x2, y2}, {x3, y3}}, PlotStyle → {PointSize → Large}];
p2 = Plot[L3[x, x0, x1, x2, x3, y0, y1, y2, y3], {x, x0, x3}];
Show[p2, p1]
```

Out[401]=
$$\frac{2}{27}(4-x)(10-x)x-\frac{1}{24}(10-x)(-1+x)x-\frac{1}{540}(-4+x)(-1+x)x$$

Out[404]=



Exercise 8

8. Use interpolating polynomials of degrees one, two, and three to approximate each of the following:

(a) $f(0.43)$ if $f(0) = 1$, $f(0.25) = 1.64872$, $f(0.5) = 2.71828$, $f(0.75) = 4.48169$.

(b) $f(0.18)$ if $f(0.1) = -0.29004986$, $f(0.2) = -0.56079734$, $f(0.3) = -0.81401972$, $f(0.4) = -1.0526302$. [S]

(c) $f(2.26)$ if $f(1) = 1.654$, $f(1.5) = -2.569$, $f(2) = -1.329$, $f(2.5) = 1.776$. [S]

(d) $f(11.26)$ if $f(10) = -0.7865$, $f(11) = -1.2352$, $f(12) = -0.8765$, $f(13) = 0.0021$.

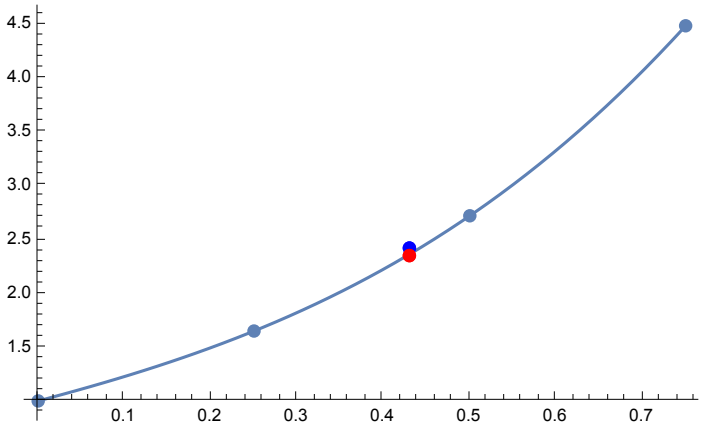
In[405]:=
```
a = 0.43;
{x0, x1, x2, x3} = {0, 0.25, 0.5, 0.75};
{y0, y1, y2, y3} = {1, 1.64872, 2.71828, 4.48169};
fa1 = L1[a, x1, x2, y1, y2]
fa2 = L2[a, x1, x2, x3, y1, y2, y3]
fa3 = L3[a, x0, x1, x2, x3, y0, y1, y2, y3]
p1 =
  ListPlot[{{x0, y0}, {x1, y1}, {x2, y2}, {x3, y3}}, PlotStyle → {PointSize → Large}];
p1a = ListPlot[{{a, fa1}}, PlotStyle → {Blue, PointSize → Large}];
p1b = ListPlot[{{a, fa2}}, PlotStyle → {Red, PointSize → Large}];
p1c = ListPlot[{{a, fa3}}, PlotStyle → {Yellow, PointSize → Large}];
p2 = Plot[L3[x, x0, x1, x2, x3, y0, y1, y2, y3], {x, x0, x3}];
Show[p2, p1, p1a, p1b]
```

Out[408]= 2.4188032

Out[409]= 2.34886312

Out[410]= 2.36060473408

Out[416]=



Let's try to be smart about this and write a function to do each:

In[417]:=

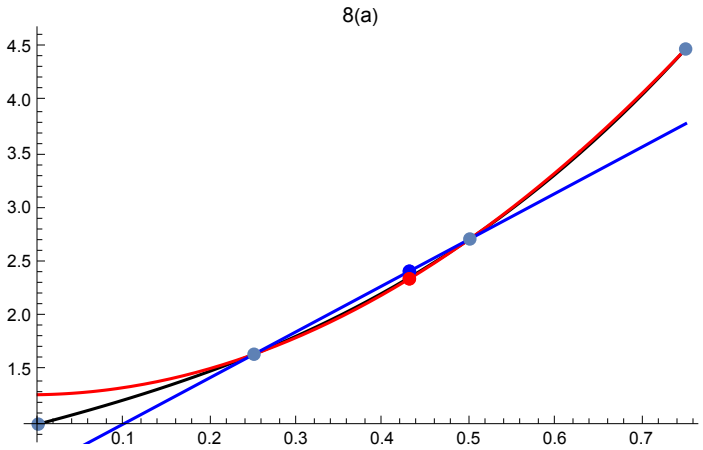
```
Ex8[a_, xpoints_, ypoints_, title_] :=
 Module[{
   x0 = xpoints[[1]], x1 = xpoints[[2]], x2 = xpoints[[3]], x3 = xpoints[[4]],
   y0 = ypoints[[1]], y1 = ypoints[[2]], y2 = ypoints[[3]], y3 = ypoints[[4]],
   fa1, fa2, fa3, p1, p1a, p1b, p2, p3, p4
  },
  fa1 = L1[a, x0, x1, y0, y1];
  fa2 = L2[a, x0, x1, x2, y0, y1, y2];
  fa3 = L3[a, x0, x1, x2, x3, y0, y1, y2, y3];
  Print[{fa1, fa2, fa3}];
  p1 = ListPlot[
    {{x0, y0}, {x1, y1}, {x2, y2}, {x3, y3}}, PlotStyle → {PointSize → Large}];
  p1a = ListPlot[{{a, fa1}}, PlotStyle → {Blue, PointSize → Large}];
  p1b = ListPlot[{{a, fa2}}, PlotStyle → {Red, PointSize → Large}];
  p2 =
   Plot[L1[x, x0, x1, y0, y1], {x, Min[xpoints], Max[xpoints]}, PlotStyle → Blue];
  p3 = Plot[L2[x, x0, x1, x2, y0, y1, y2],
    {x, Min[xpoints], Max[xpoints]}, PlotStyle → Red];
  p4 = Plot[L3[x, x0, x1, x2, x3, y0, y1, y2, y3],
    {x, Min[xpoints], Max[xpoints]}, PlotStyle → Black, PlotLabel → title];
  Show[p4, p3, p2, p1, p1a, p1b]
 ]
```

In[418]:=

```
a = 0.43;
xpoints = {0.25, 0.5, 0.75, 0};
ypoints = {1.64872, 2.71828, 4.48169, 1};
Ex8[a, xpoints, ypoints, "8(a)"]
```

```
{2.4188032, 2.34886312, 2.36060473408}
```

Out[421]=

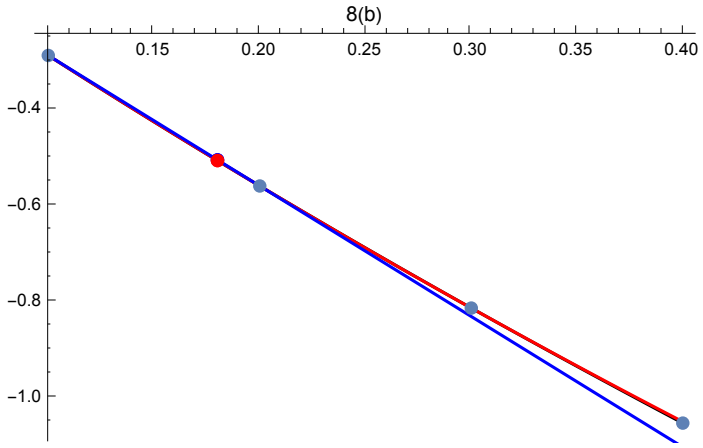


In[422]:=

```
a = 0.18;
xpoints = {0.1, 0.2, 0.3, 0.4};
ypoints = {-0.29004986, -0.56079734, -0.81401972, -1.0526302};
Ex8[a, xpoints, ypoints, "8(b)"]
```

```
{-0.506647844, -0.508049852, -0.5081430744}
```

Out[425]=

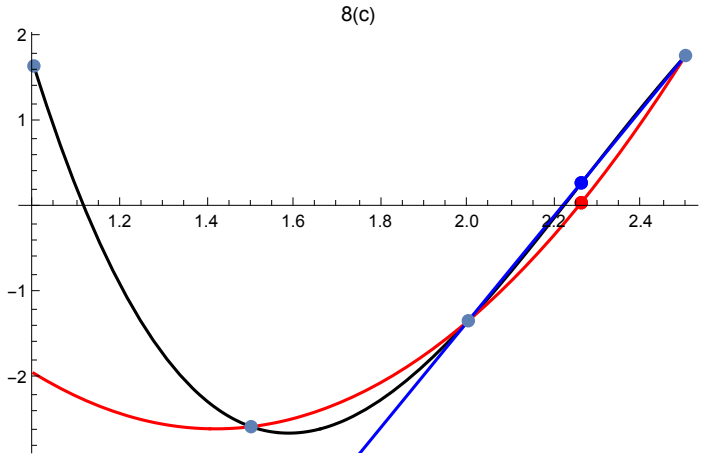


In[426]:=

```
a = 2.26;
xpoints = {2, 2.5, 1.5, 1};
ypoints = {-1.329, 1.776, -2.569, 1.654};
Ex8[a, xpoints, ypoints, "8(c)"]
```

```
{0.285599999999999, 0.0528479999999987, 0.280356735999998}
```

Out[429]=

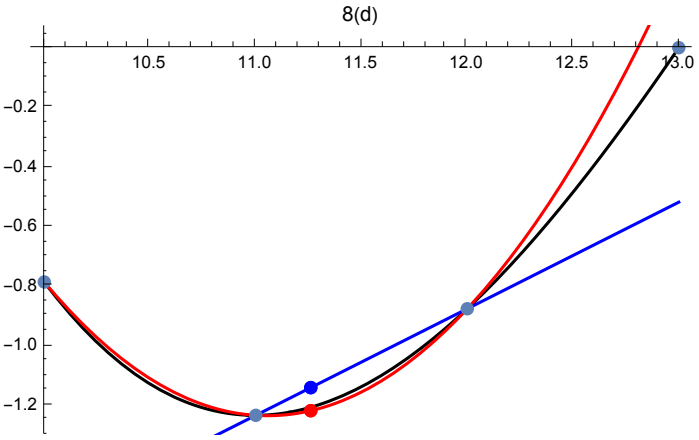


In[430]:=

```
a = 11.26;
xpoints = {11, 12, 10, 13};
ypoints = {-1.2352, -0.8765, -0.7865, 0.0021};
Ex8[a, xpoints, ypoints, "8(d)"]
```

{-1.141938, -1.21960988, -1.20799373}

Out[433]=



Exercise 9

In[434]:=
```
Ex9[a_, true_, xpoints_, ypoints_, title_] :=
 Module[{
   x0 = xpoints[[1]], x1 = xpoints[[2]], x2 = xpoints[[3]],
   y0 = ypoints[[1]], y1 = ypoints[[2]], y2 = ypoints[[3]],
   fa1, fa2, fa3, p1, p1a, p1b, p2, p3, p4
  },
  fa1 = L1[a, x0, x1, y0, y1];
  Print[Simplify[L1[x, x0, x1, y0, y1]]];
  fa2 = L2[a, x0, x1, x2, y0, y1, y2];
  Print[Simplify[L2[x, x0, x1, x2, y0, y1, y2]]];
  Print[{fa1, fa2}];
  Print[Abs[true - {fa1, fa2}]];
  p1 = ListPlot[{{x0, y0}, {x1, y1}, {x2, y2}}, PlotStyle → {PointSize → Large}];
  p1a = ListPlot[{{a, fa1}}, PlotStyle → {Blue, PointSize → Large}];
  p1b = ListPlot[{{a, fa2}}, PlotStyle → {Red, PointSize → Large}];
  p2 =
   Plot[L1[x, x0, x1, y0, y1], {x, Min[xpoints], Max[xpoints]}, PlotStyle → Blue];
  p3 = Plot[L2[x, x0, x1, x2, y0, y1, y2],
    {x, Min[xpoints], Max[xpoints]}, PlotStyle → Red, PlotLabel → title];
  Show[p3, p2, p1, p1a, p1b, PlotRange → All]
 ]
```

In[435]:=
```
a = 1.4;
xpoints = {1.25, 1.6, 1};
f[x_] := Sin[Pi x]
ypoints = f[xpoints];
Ex9[a, f[a], xpoints, ypoints, "9(a)"]
```

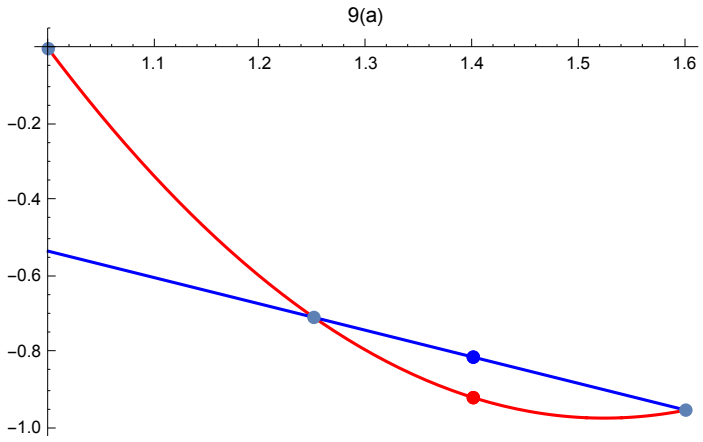
$0.164142272772761 - 0.696999243167447\,x$

$7.26890187803524 - 10.8212816806665\,x + 3.55237980263124\,x^2$

$\{-0.811656667661664, -0.918228061740601\}$

$\{0.139399848633489, 0.0328284545545522\}$

Out[439]=



In[440]:=
```
f[x_] := CubeRoot[x - 1]
ypoints = f[xpoints];
Ex9[a, f[a], xpoints, ypoints, "9(b)"]
```

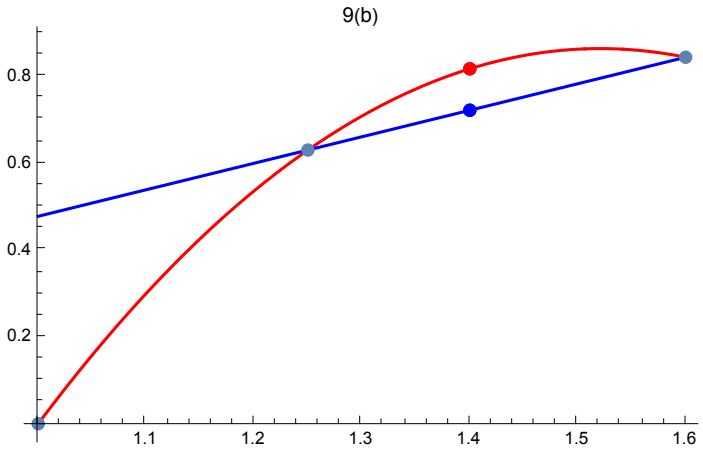
$-0.132439976317966 + 0.609920401012322\,x$

$-6.49884563890938 + 9.68204847020509\,x - 3.18320283129571\,x^2$

$\{0.721448585099285, 0.816944670038156\}$

$\{0.0153577146287925, 0.0801383703100786\}$

Out[442]=



In[443]:=
```
f[x_] := E^(2 x - 4)
ypoints = f[xpoints];
Ex9[a, f[a], xpoints, ypoints, "9(c)"]
```

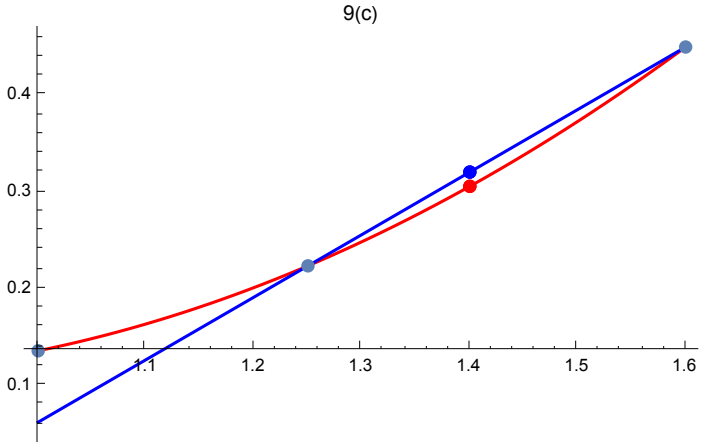
$-0.584722711168684 + 0.646282297053691\ x$

$0.398953253519392 - 0.755455952626816\ x + 0.491837982344038\ x^2$

{0.320072504706483, 0.305317365236162}

{0.0188782927942814, 0.00412315332396024}

Out[445]=



In[446]:=
```
f[x_] := Log[10 x]
ypoints = f[xpoints];
Ex9[a, f[a], xpoints, ypoints, "9(d)"]
```

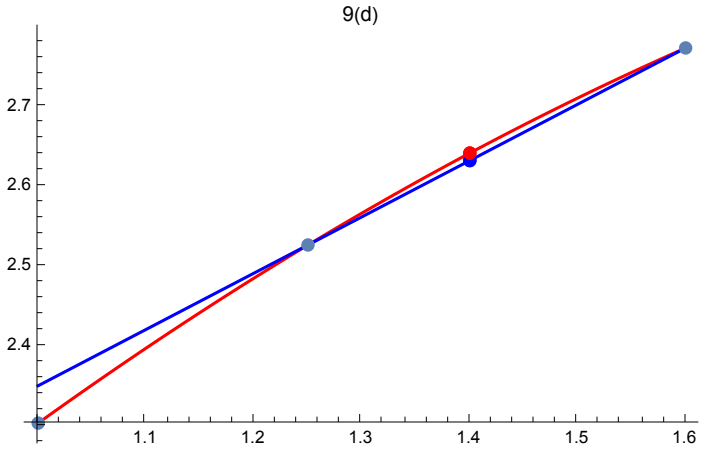
$1.64408550883852 + 0.705314508375787\ x$

$1.01988651923502 + 1.59479806856078\ x - 0.312099494801752\ x^2$

{2.63152582056462, 2.64088880540868}

{0.00753150905063515, 0.00183147579341769}

Out[448]=



Exercise 10

10. Use formula 3.2.3 to find theoretical error bounds for the approximations in question 9. Compare the bound to the actual error. [S]

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n).$$

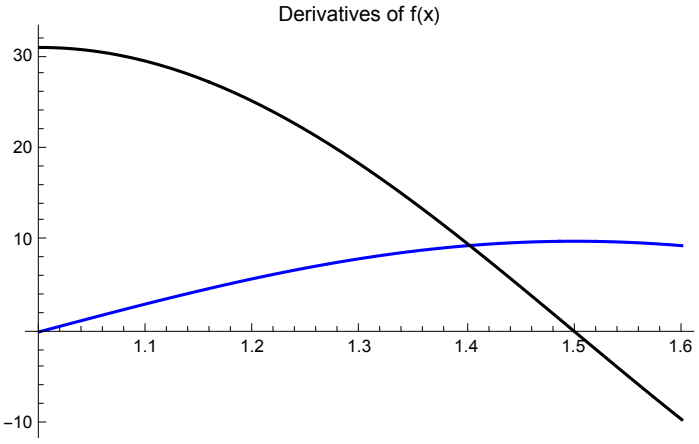
In[449]:=
```
prodlist[a_, list_] := Times @@ (a - list)
prodlist[3, {3.1, 6, 4}]
```

Out[450]=
```
-0.3
```

In[451]:=
```
Ex10[a_, f_, b_, c_, xpoints_, ypoints_, title_] :=
 Module[{
   x0 = xpoints[[1]], x1 = xpoints[[2]], x2 = xpoints[[3]],
   y0 = ypoints[[1]], y1 = ypoints[[2]], y2 = ypoints[[3]],
   fa1, fa2, fa3, p1, p1a, p1b, p2, p3, p4,
   fpp = f''[a], fppp = f'''[a], product = prodlist[a, xpoints]
  },
  fa1 = L1[a, x0, x1, y0, y1];
  fa2 = L2[a, x0, x1, x2, y0, y1, y2];
  Print[{fa1, fa2}];
  Print[Abs[f[a] - {fa1, fa2}]];
  Print[Abs[{f''[b], f'''[c] / 3 * (a - x2)} / 2! * (a - x0) * (a - x1)]];
  p1 = ListPlot[{{x0, y0}, {x1, y1}, {x2, y2}}, PlotStyle → {PointSize → Large}];
  p1a = ListPlot[{{a, fa1}}, PlotStyle → {Blue, PointSize → Large}];
  p1b = ListPlot[{{a, fa2}}, PlotStyle → {Red, PointSize → Large}];
  p2 =
   Plot[L1[x, x0, x1, y0, y1], {x, Min[xpoints], Max[xpoints]}, PlotStyle → Blue];
  p3 = Plot[L2[x, x0, x1, x2, y0, y1, y2],
    {x, Min[xpoints], Max[xpoints]}, PlotStyle → Red];
  p4 = Plot[f[x], {x, Min[xpoints], Max[xpoints]},
    PlotStyle → Black, PlotLabel → title];
  Show[p4, p3, p2, p1, p1a, p1b, PlotRange → All]
 ]
```

In[452]:=
```
a = 1.4;
xpoints = {1.25, 1.6, 1};
f[x_] := Sin[Pi x]
ypoints = f[xpoints];
Plot[{f''[x], f'''[x]}, {x, Min[xpoints], Max[xpoints]},
 PlotStyle → {Blue, Black}, PlotLabel → "Derivatives of f(x)"]
f[a]
b = 3 / 2;
c = 1;
Ex10[a, f, b, c, xpoints, ypoints, "10(a)"]
```

Out[456]=



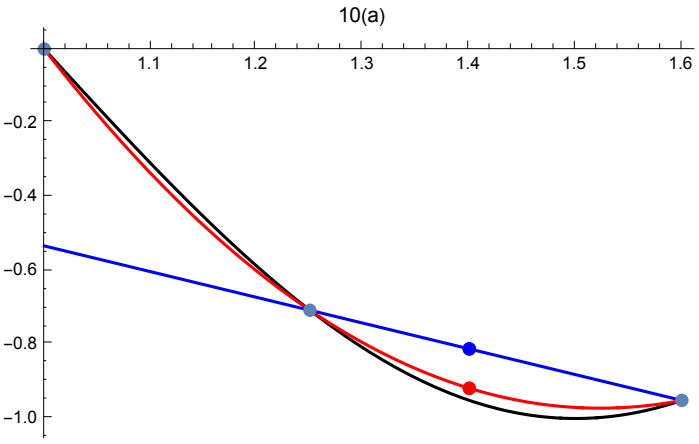
Out[457]= −0.951056516295154

{−0.811656667661664, −0.918228061740601}

{0.139399848633489, 0.0328284545545522}

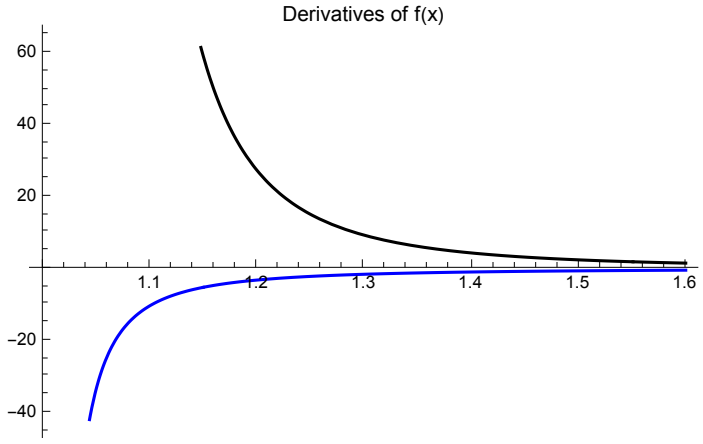
{0.14804406601634, 0.0620125533605996}

Out[460]=



In[461]:=
```
a = 1.4;
xpoints = {1.25, 1.6, 1};
f[x_] := CubeRoot[x - 1]
ypoints = f[xpoints];
Plot[{f''[x], f'''[x]}, {x, Min[xpoints], Max[xpoints]},
 PlotStyle → {Blue, Black}, PlotLabel → "Derivatives of f(x)"]
f[a]
b = 1.25;
c = 1;
Ex10[a, f, b, c, xpoints, ypoints, "10(b)"]
```

Out[465]=



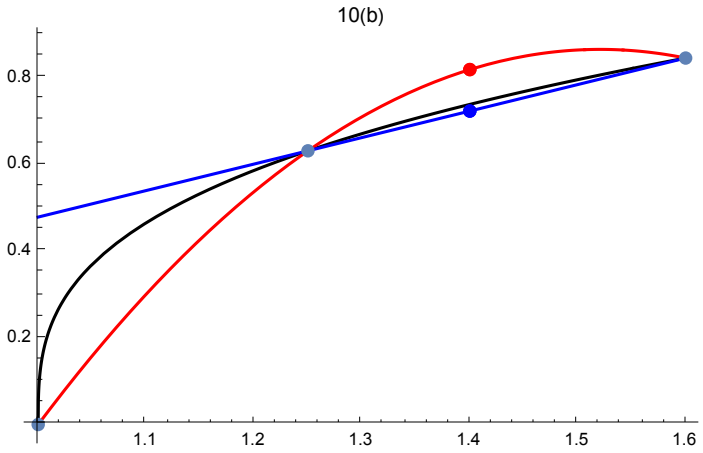
Out[466]= 0.736806299728077

{0.721448585099285, 0.816944670038156}

{0.0153577146287925, 0.0801383703100786}

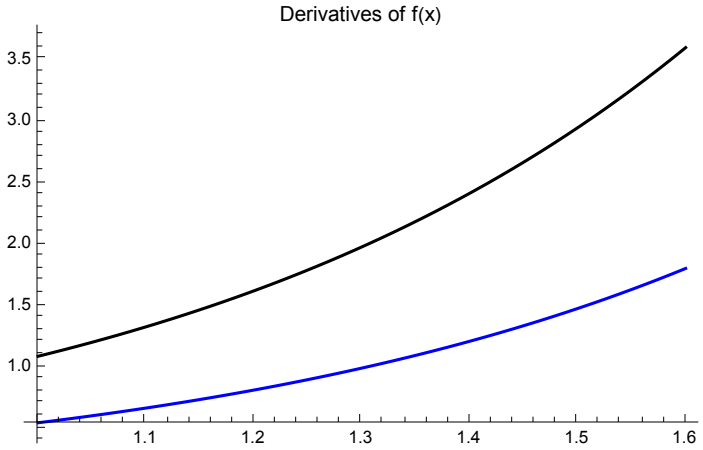
{0.0335978946638633, ∞}

Out[469]=



In[470]:=
```
a = 1.4;
xpoints = {1.25, 1.6, 1};
f[x_] := E^(2 x - 4)
ypoints = f[xpoints];
Plot[{f''[x], f'''[x]}, {x, Min[xpoints], Max[xpoints]},
 PlotStyle → {Blue, Black}, PlotLabel → "Derivatives of f(x)"]
f[a]
b = 1.6;
c = 1.6;
Ex10[a, f, b, c, xpoints, ypoints, "10(c)"]
```

Out[474]=



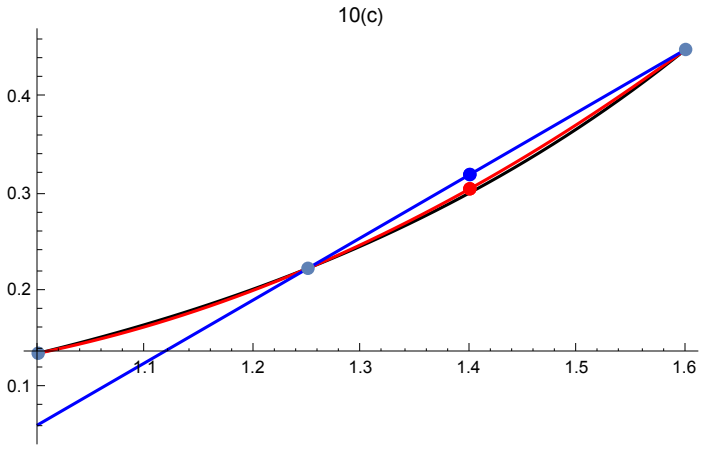
Out[475]= 0.301194211912202

{0.320072504706483, 0.305317365236162}

{0.0188782927942814, 0.00412315332396024}

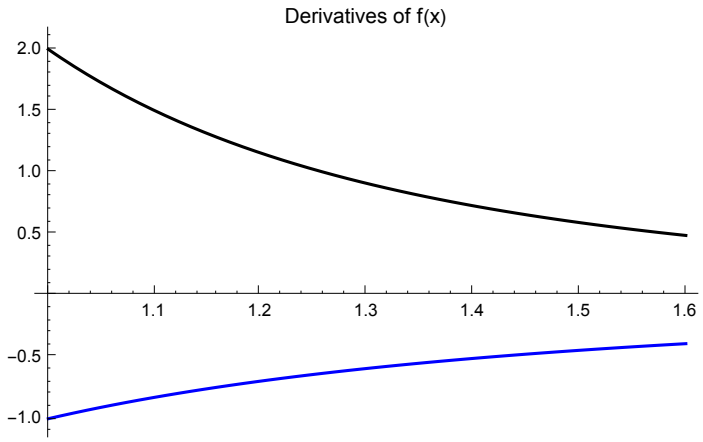
{0.0269597378470333, 0.00718926342587555}

Out[478]=



In[479]:=
```
a = 1.4;
xpoints = {1.25, 1.6, 1};
f[x_] := Log[10 x]
ypoints = f[xpoints];
Plot[{f''[x], f'''[x]}, {x, Min[xpoints], Max[xpoints]},
 PlotStyle → {Blue, Black}, PlotLabel → "Derivatives of f(x)"]
f[a]
b = 1.25;
c = 1;
Ex10[a, f, b, c, xpoints, ypoints, "10(d)"]
```

Out[483]=



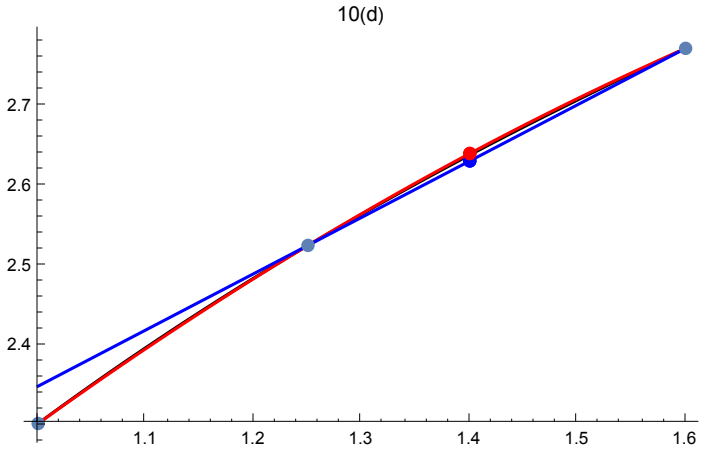
Out[484]= 2.63905732961526

{2.63152582056462, 2.64088880540868}

{0.00753150905063515, 0.00183147579341769}

{0.0096, 0.004}

Out[487]=



Exercise 13

Recursive definition of Interpolating polynomials: Neville's method

Neville's method: A method for computing the interpolating polynomial of least degree or values of it based on the recursive relation

$$\begin{aligned} P_{i,m+1}(x) &= \frac{(x - x_i)P_{i+1,m}(x) - (x - x_{i+m+1})P_{i,m}(x)}{x_{i+m+1} - x_i} \\ P_{i,0}(x) &= f(x_i) \end{aligned}$$

where $P_{k,l}$ is the polynomial of least degree interpolating the data

$$(x_k, f(x_k)), (x_{k+1}, f(x_{k+1})), \ldots, (x_{k+l}, f(x_{k+l})).$$

Neville's method is sort of the ideal limit of one of my favorite principles: that whenever you have two estimates for a quantity, you have a third: the average. In this case, the average of successive averages actually gives you the exact value at the end! We just slowly work up to the answer.

Each successive p[[i,j]] is a weighted average estimate of f(x), using the linear polynomial through two points.

In[488]:=
```
nevillesMethod[x_, data_, verbose_] :=
 Module[{n = Length[data], xs, ys, i, j, p},
  p = ConstantArray[0, {n, n}];
  {xs, ys} = Transpose[data];  (* The data is a list of coordinates *)
  p[[All, 1]] = ys; (* We initial the first column of p with the y-values *)
  For[j = 2, j ≤ n + 1, j++,
   For[i = 1, i ≤ n + 1 - j, i++,
     p[[i, j]] = (* Success columns are filled with interpolated value,
       weighted linear interpolators of averaged y-
        values from the previous column. *)
       ((x - xs[[i]]) p[[i + 1, j - 1]] - (x - xs[[i + j - 1]]) p[[i, j - 1]])
        /
        (xs[[i + j - 1]] - xs[[i]]);
    ];
  ];
  If[verbose, Print[MatrixForm[p]]];
  p[[1, n]]
 ]
```

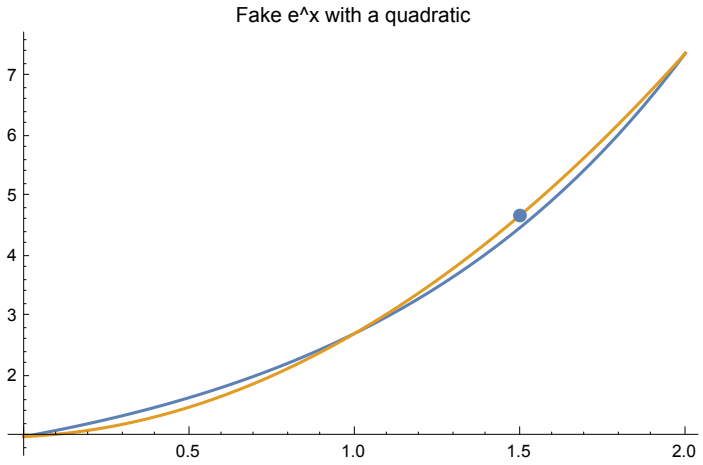
Table 3.2, p. 119: Example of Neville's method

In[489]:=
```
a = 1.5;
fapprox = nevillesMethod[a, {{0, 1}, {1, E}, {2, E^2}}, True]
fakeExp[x_] := nevillesMethod[x, {{0, 1}, {1, E}, {2, E^2}}, False]
Show[Plot[{E^x, fakeExp[x]}, {x, 0, 2}, PlotLabel → "Fake e^x with a quadratic"],
 ListPlot[{{a, fapprox}}, PlotStyle → {PointSize → Large}]
]
```

$$\begin{pmatrix} 1 & 3.57742274268857 & 4.68460740844328 \\ e & 5.05366896369485 & 0 \\ e^2 & 0 & 0 \end{pmatrix}$$

Out[490]= 4.68460740844328

Out[492]=

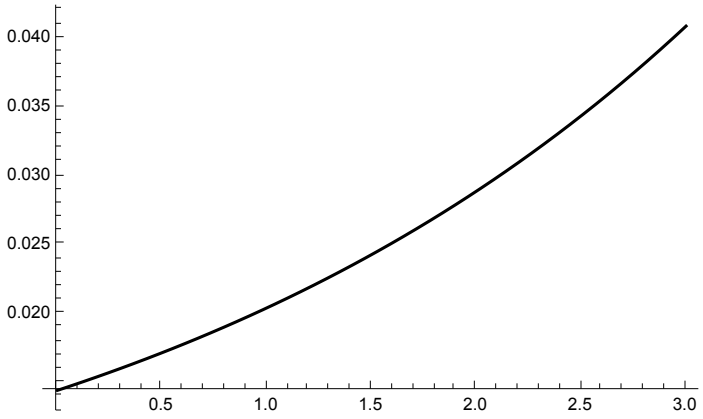


In[493]:=
```
a = 1.5;
f[x_] := 2^(x / 2)
xpoints = {0, 1, 2, 3};
fapprox = nevillesMethod[a, Transpose[{xpoints, f[xpoints]}], True]
Plot[{f''''[x]}, {x, Min[xpoints], Max[xpoints]}, PlotStyle → {Blue, Black}]
f''''[3] / 4! * prodlist[a, xpoints]
Abs[f[a] - fapprox]
```

$$\begin{pmatrix} 1 & 1.62132034355964 & 1.68566017177982 & 1.68121843353823 \\ \sqrt{2} & 1.70710678118655 & 1.67677669529664 & 0 \\ 2 & 1.5857864376269 & 0 & 0 \\ 2\sqrt{2} & 0 & 0 & 0 \end{pmatrix}$$

Out[496]= 1.68121843353823

Out[497]=



Out[498]= 0.000956396856702911

Out[499]= 0.000574396969200031

Exercise 22

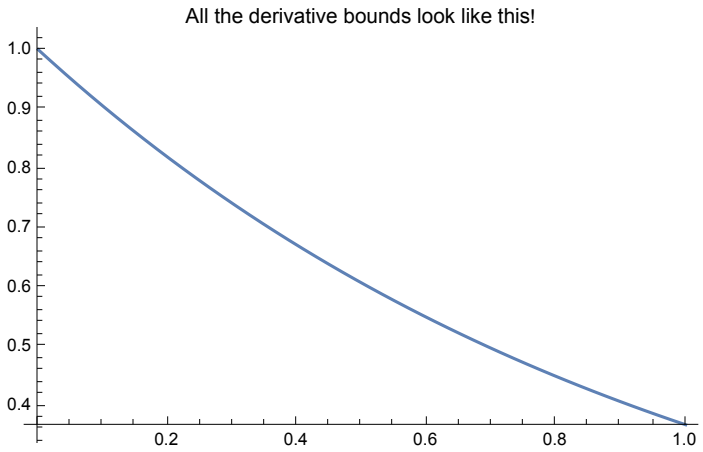
22. Let $f(x) = e^{-x}$. Two different numbers are chosen at random from the interval $[0, 1]$, say x_0 and x_1. Then the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ are used to get a linear Lagrange interpolation approximation to f over the interval $[0, 1]$. Find a bound (good for the entire interval and every pair of points x_0 and x_1) for the error in using this approximation.

In[500]:=
```
f[x_] := E^(-x)
```

The absolute value of any derivative is going to be maximized at x=0, and equals 1:

In[501]:=
```
Plot[f''[x], {x, 0, 1}, PlotLabel → "All the derivative bounds look like this!"]
```

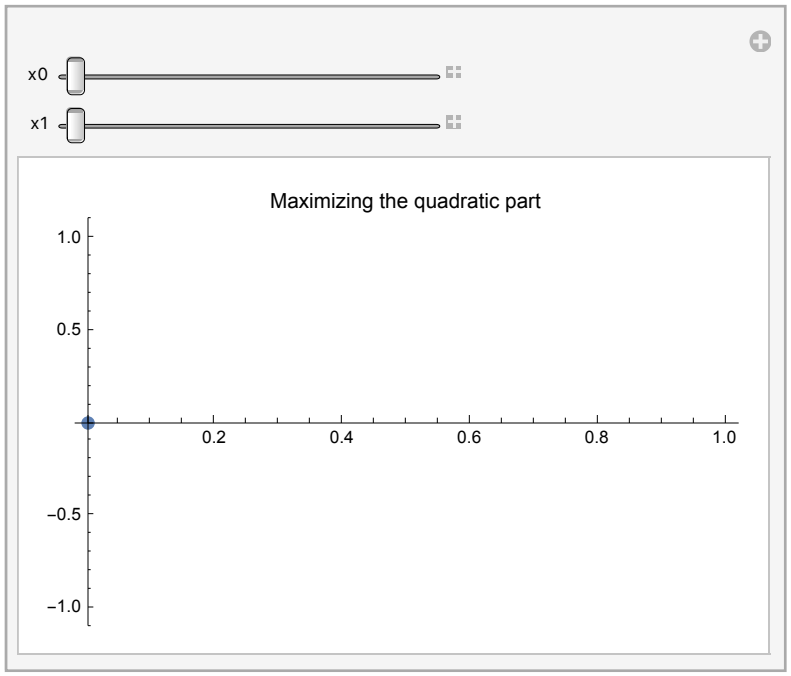
Out[501]=



Picking any two points on the interval [0,1], we examine the function (x-x0)(x-x1). If both were chosen to be the same endpoint (e.g. x0=x1=0), then the function is x^2, and is maximized at 1, with value 1. That is the maximum possible.

In[502]:=
```
Manipulate[
 Show[
  Plot[(x - x0) (x - x1), {x, 0, 1}, PlotLabel → "Maximizing the quadratic part"],
  ListPlot[{{x0, 0}, {x1, 0}}, PlotStyle → {PointSize → Large}]
 ],
 {x0, 0, x1},
 {x1, x0, 1}
]
```

Out[502]=



Therefore the error bound is (1)/2!*1=1/2

In[503]:=
```
1 / 2.0
```

Out[503]=
```
0.5
```

Worst case scenarios involve using a tangent at either endpoint, and the absolute errors are within our bound (comfortably):

In[504]:=
```
Plot[{f[x], 1 - x, E^(-1) - E^(-1) (x - 1)},
 {x, 0, 1}, PlotLabel → "with worst case scenarios...."]
```

Out[504]=