Homework 3.3: Numerical Calculus

Andy Long, Spring, 2024

#1 and 2, p. 129

1. Modify the Neville's method pseudo-code on page 119 to produce pseudo-code for computing the coefficients of N_n.
2. Modify the Neville's method Octave code on page 120 to produce octave code for computing the coefficients of N_n. Test it by computing N_2 interpolating $f(x) = e^x$ at $x = 0, 1, 2$ and comparing your result to that on page 124.

#1. Newton Polynomials: Divided differences is how we compute the coefficients of the Newton Polynomial:

Define divided differences:

Definition [edit]

Given $n + 1$ data points

$$(x_0, y_0), \ldots, (x_n, y_n)$$

where the x_k are assumed to be pairwise distinct, the **forward divided differences** are defined as:

$$\begin{aligned} [y_k] &:= y_k, & k &\in \{0, \ldots, n\} \\ [y_k, \ldots, y_{k+j}] &:= \frac{[y_{k+1}, \ldots, y_{k+j}] - [y_k, \ldots, y_{k+j-1}]}{x_{k+j} - x_k}, & k &\in \{0, \ldots, n-j\},\ j \in \{1, \ldots, n\}. \end{aligned}$$

In[3628]:=
```
dividedDifference[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 initialize 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]] = (* Successive columns are filled with divided differences,
       from the previous column. *)
       (p[[i + 1, j - 1]] - p[[i, j - 1]]) / (xs[[i + j - 1]] - xs[[i]]);
    ];
  ];
  If[verbose, Print[MatrixForm[p]]];
  p[[1, All]]
 ]
```

#2. Let's test out our code:

In[3629]:=
```
xpoints = {0.0, 1.0, 2.0};
ypoints = E^xpoints;
newtonCoefs = dividedDifference[Transpose[{xpoints, ypoints}], True]
```

$$\begin{pmatrix} 1. & 1.71828182845905 & 1.47624622100628 \\ 2.71828182845905 & 4.67077427047161 & 0 \\ 7.38905609893065 & 0 & 0 \end{pmatrix}$$

Out[3631]= {1., 1.71828182845905, 1.47624622100628}

It appears to be working. At least it matches what's given on p. 124.

This will be useful: Hornerize the Newton Polynomial

In[3632]:=
```
interpolator[x_, coefs_, xs_, verbose_] :=
 Module[
  {result, n = Length[xs]},
  result = coefs[[n]];
  For[i = n - 1, i > 0, i--,
   If[verbose, Print[{xs[[i]], coefs[[i]]}]];
   result = coefs[[i]] + (x - xs[[i]]) result;
  ];
  result
 ]
```

#3, p. 129

3. Let $f(0.1) = 0.12$, $f(0.2) = 0.14$, $f(0.3) = 0.13$, and $f(0.4) = 0.15$.

   (a) Find the leading coefficient of the polynomial of least degree interpolating these data.

   (b) Suppose, additionally, that $f(0.5) = 0.11$. Use your previous work to find the leading coefficient of the polynomial of least degree interpolating all of the data.

In[3633]:=
```
newtonCoefs = dividedDifference[
  {{0.1, 0.12}, {0.2, 0.14}, {0.3, 0.13}, {0.4, 0.15}, {0.5, 0.11}}, True]
```

$$\begin{pmatrix} 0.12 & 0.2 & -1.5 & 10. & -62.5 \\ 0.14 & -0.1 & 1.5 & -15. & 0 \\ 0.13 & 0.2 & -3. & 0 & 0 \\ 0.15 & -0.4 & 0 & 0 & 0 \\ 0.11 & 0 & 0 & 0 & 0 \end{pmatrix}$$

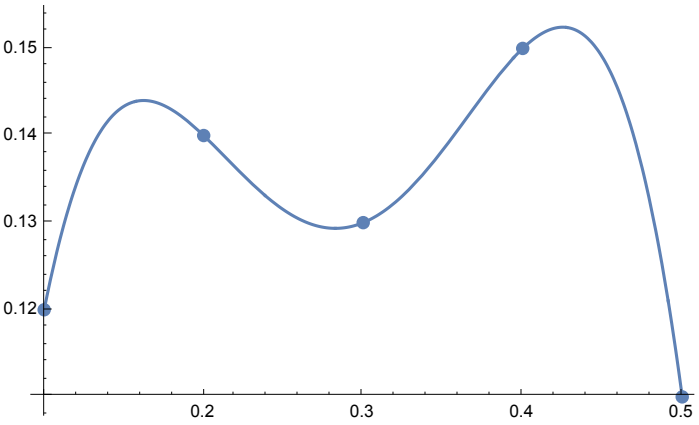
Out[3633]= {0.12, 0.2, -1.5, 10., -62.5}

In[3634]:=
```
data = {{0.1, 0.12}, {0.2, 0.14}, {0.3, 0.13}, {0.4, 0.15}, {0.5, 0.11}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 0.12 & 0.2 & -1.5 & 10. & -62.5 \\ 0.14 & -0.1 & 1.5 & -15. & 0 \\ 0.13 & 0.2 & -3. & 0 & 0 \\ 0.15 & -0.4 & 0 & 0 & 0 \\ 0.11 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Out[3636]= {0.12, 0.2, -1.5, 10., -62.5}

Out[3637]=



#4, p. 129

4. Find a Newton form of the polynomial of degree at most 3 interpolating the points $(1,2)$, $(2,2)$, $(3,0)$ and $(4,0)$. [S]

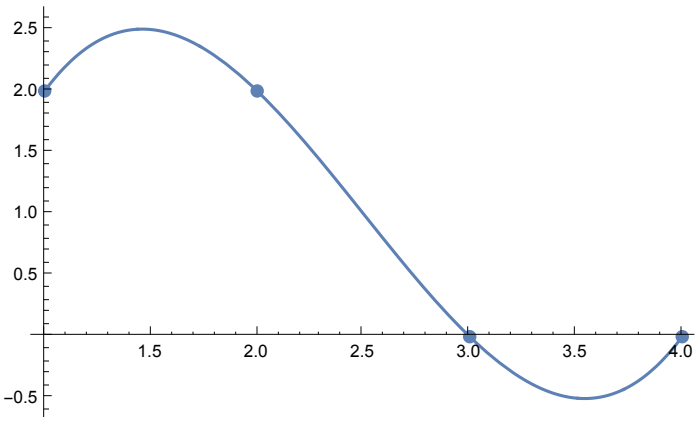
(solution p. 277 (288))

In[3638]:=
```
data = {{1, 2}, {2, 2}, {3, 0}, {4, 0}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 2 & 0 & -1 & \frac{2}{3} \\ 2 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Out[3640]= $\left\{2, 0, -1, \frac{2}{3}\right\}$

Out[3641]=



#5, p. 129

5. Use the method of divided differences to find the at-most-second-degree polynomial interpolating the points $(0, 10)$, $(30, 58)$, $(1029, -32)$. [A]

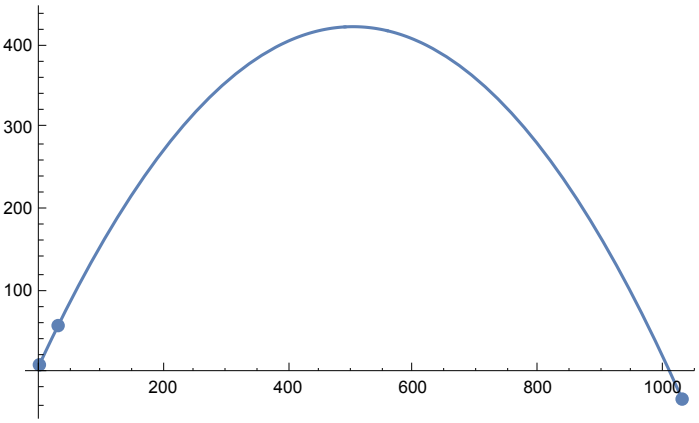
(solution p. 277 (288))

In[3642]:=
```
data = {{0, 10}, {30, 58}, {1029, -32.0}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 10 & \frac{8}{5} & -0.00164245878531593 \\ 58 & -0.0900900900900901 & 0 \\ -32. & 0 & 0 \end{pmatrix}$$

Out[3644]= $\left\{10, \frac{8}{5}, -0.00164245878531593\right\}$

Out[3645]=



#6, p. 129

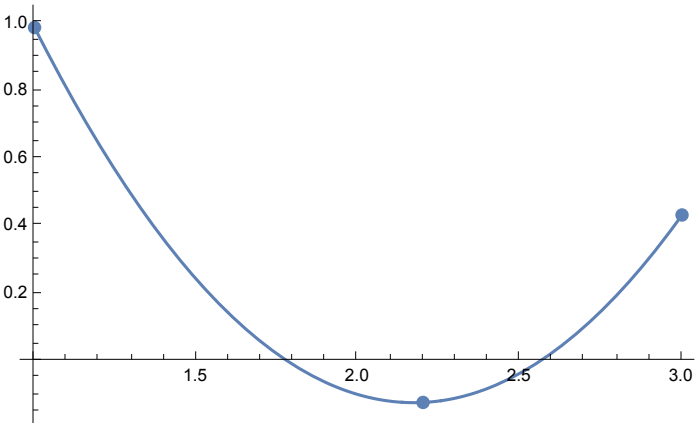
6. Use divided differences to find an interpolating polynomial for the data $f(1) = 0.987$, $f(2.2) = -0.123$, and $f(3) = 0.432$. [5]

In[3646]:=
```
data = {{1, 0.987}, {2.2, -0.123}, {3, 0.432}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 0.987 & -0.925 & 0.809375 \\ -0.123 & 0.69375 & 0 \\ 0.432 & 0 & 0 \end{pmatrix}$$

Out[3648]= {0.987, -0.925, 0.809375}

Out[3649]=



#7, p. 129

7. Create a divided differences table for the following data using only pencil and paper.

$$f(1.2) = 2.2 \quad f(1.4) = 2.1 \quad f(1.6) = 2.3$$

(a) What is the interpolating polynomial of degree at most 2? Does it actually have degree 2?

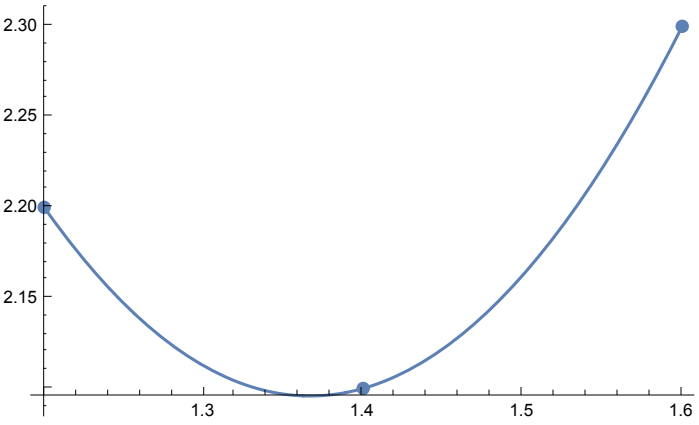
(b) Write down two distinct linear interpolating polynomials for this data based on your table.

In[3650]:=
```
data = {{1.2, 2.2}, {1.4, 2.1}, {1.6, 2.3}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 2.2 & -0.500000000000001 & 3.74999999999999 \\ 2.1 & 0.999999999999998 & 0 \\ 2.3 & 0 & 0 \end{pmatrix}$$

Out[3652]= {2.2, -0.500000000000001, 3.74999999999999}

Out[3653]=



#8, p. 129

8. Use divided differences to find the at-most-cubic polynomial of exercise 19 of section 3.2. Does it have the expected degree? [A]

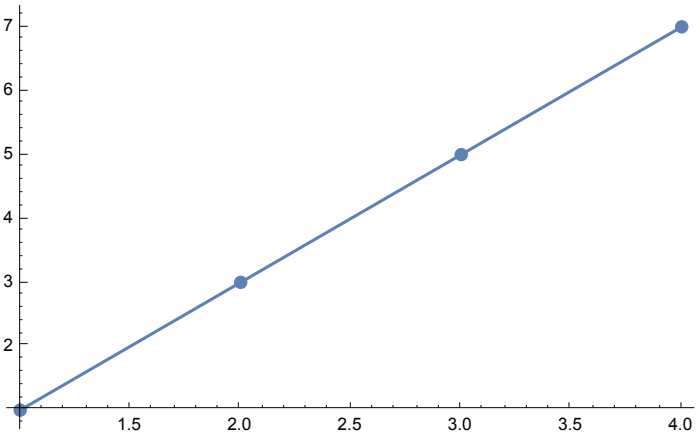
19. The interpolating polynomial on $n+1$ points does not always have degree n. It has degree at most n. Plot the data $(1,1)$, $(2,3)$, $(3,5)$, and $(4,7)$, and make a conjecture as to the degree of the polynomial interpolating these four points. What led you to your conjecture?

In[3654]:=
```
data = {{1, 1}, {2, 3}, {3, 5}, {4, 7}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 7 & 0 & 0 & 0 \end{pmatrix}$$

Out[3656]= {1, 2, 0, 0}

Out[3657]=



Well, looking at the data, the expected degree is 1 -- so it depends on what one means by “expected”...:)

#9, p. 129

9. Find the degree at most two interpolating polynomial of the form

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

for the data in the table.

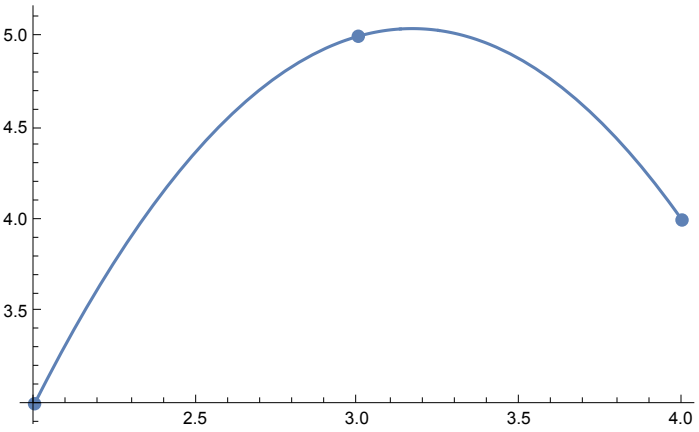
i	0	1	2
x_i	2	3	4
$f(x_i)$	3	5	4

In[3658]:=
```
data = {{2, 3}, {3, 5}, {4, 4}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 3 & 2 & -\frac{3}{2} \\ 5 & -1 & 0 \\ 4 & 0 & 0 \end{pmatrix}$$

Out[3660]= $\left\{3, 2, -\frac{3}{2}\right\}$

Out[3661]=



#10, p. 129

10. Use the Octave code from question 2 to compute the interpolating polynomial of at most degree four for the data:

x	$f(x)$
0.0	−6.00000
0.1	−5.89483
0.3	−5.65014
0.6	−5.17788
1.0	−4.28172

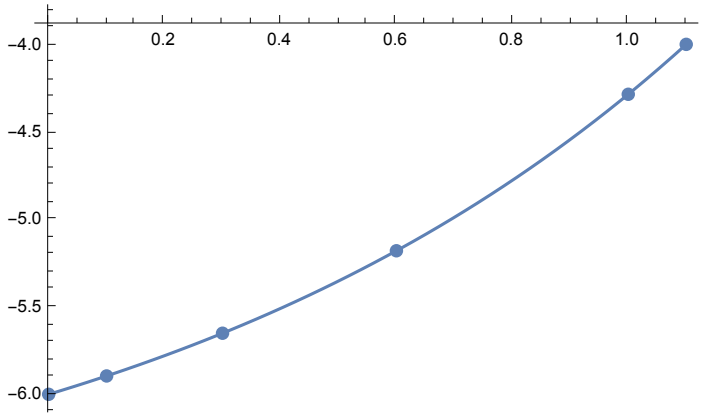
Then add $f(1.1) = -3.9958$ to the table, and compute the interpolating polynomial of degree at most 5 using a calculator. You may use the Octave code to check your work. [S]

In[3662]:=
```
data = {{0.0, -6.00000}, {0.1, -5.89483},
    {0.3, -5.65014}, {0.6, -5.17788}, {1.0, -4.28172}, {1.1, -3.9958}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 PlotRange → All
]
```

```
⎛   -6.     1.0517  0.572499999999983 0.215000000000042 0.0630158730158206 0.01484126984132
⎜-5.89483 1.22345 0.701500000000008 0.278015873015862 0.0793412698412738          0
⎜-5.65014  1.5742  0.951714285714284 0.357357142857136           0                  0
⎜-5.17788  2.2404   1.23759999999999          0                  0                  0
⎜-4.28172  2.8592           0                 0                  0                  0
⎝ -3.9958     0             0                 0                  0                  0
```

Out[3664]=
```
{-6., 1.0517, 0.572499999999983,
 0.215000000000042, 0.0630158730158206, 0.014841269841321}
```

Out[3665]=



#11, p. 129

11. Use the Octave code from question 2 to find interpolating polynomials of degrees (at most) one, two, and three for the following data. Approximate $f(8.4)$ using each polynomial.

$$f(8.1) = 16.94410,\ f(8.3) = 17.56492,$$
$$f(8.6) = 18.50515,\ f(8.7) = 18.82091$$

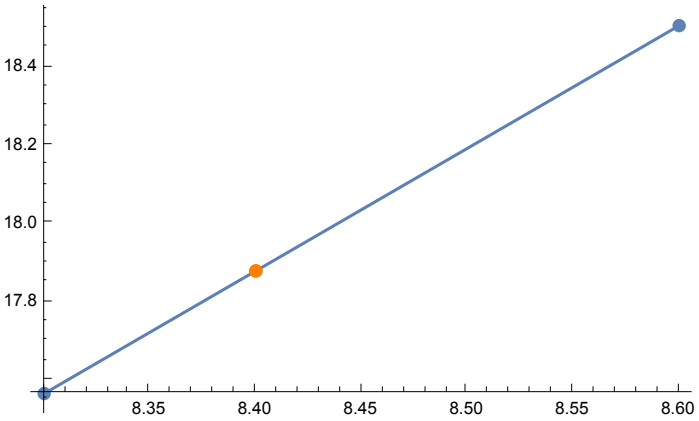
In[3666]:=
```
data = {{8.3, 17.56492}, {8.6, 18.50510}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
estimate = interpolator[8.4, newtonCoefs, xpoints, False]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 ListPlot[{{8.4, estimate}}, PlotStyle → {Orange, PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 17.56492 & 3.13393333333334 \\ 18.5051 & 0 \end{pmatrix}$$

Out[3668]= {17.56492, 3.13393333333334}

Out[3669]= 17.8783133333333

Out[3670]=



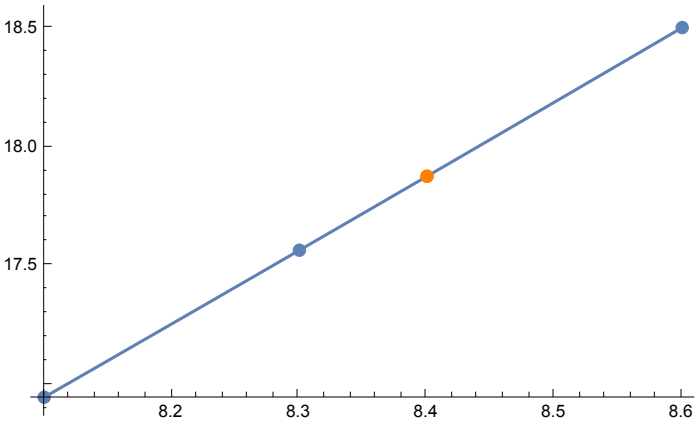
In[3671]:=
```
data = {{8.1, 16.94410}, {8.3, 17.56492}, {8.6, 18.50510}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
estimate = interpolator[8.4, newtonCoefs, xpoints, False]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 ListPlot[{{8.4, estimate}}, PlotStyle → {Orange, PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 16.9441 & 3.10409999999999 & 0.059666666666689 \\ 17.56492 & 3.13393333333334 & 0 \\ 18.5051 & 0 & 0 \end{pmatrix}$$

Out[3673]= {16.9441, 3.10409999999999, 0.059666666666689}

Out[3674]= 17.87712

Out[3675]=



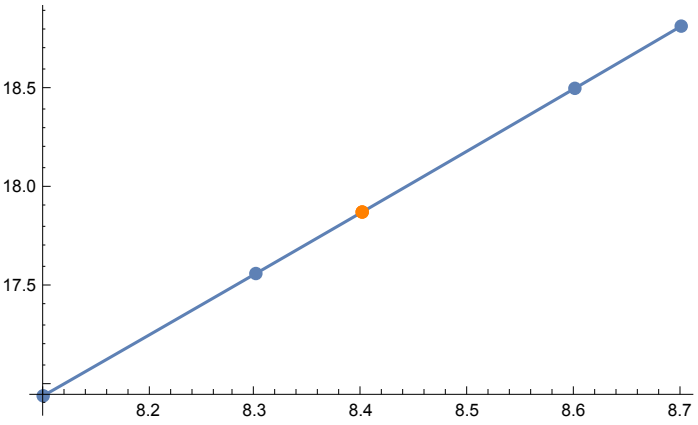
In[3676]:=
```
data = {{8.1, 16.94410}, {8.3, 17.56492}, {8.6, 18.50510}, {8.7, 18.82091}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
estimate = interpolator[8.4, newtonCoefs, xpoints, False]
Show[
 Plot[interpolator[x, newtonCoefs, xpoints, False],
  {x, Min[xpoints], Max[xpoints]}],
 ListPlot[Transpose[{xpoints, ypoints}], PlotStyle → {PointSize → Large}],
 ListPlot[{{8.4, estimate}}, PlotStyle → {Orange, PointSize → Large}],
 PlotRange → All
]
```

$$\begin{pmatrix} 16.9441 & 3.10409999999999 & 0.059666666666689 & 0.00125000000009922 \\ 17.56492 & 3.13393333333334 & 0.0604166666667485 & 0 \\ 18.5051 & 3.15810000000004 & 0 & 0 \\ 18.82091 & 0 & 0 & 0 \end{pmatrix}$$

Out[3678]= {16.9441, 3.10409999999999, 0.059666666666689, 0.00125000000009922}

Out[3679]= 17.8771125

Out[3680]=



#12, p. 129

12. Find a bound on the error in using the interpolating polynomial of question 6 to approximate $f(2)$ assuming that all derivatives of f are bounded between -2 and 1 over the interval $[1, 3]$. [S]

6. Use divided differences to find an interpolating polynomial for the data $f(1) = 0.987$, $f(2.2) = -0.123$, and $f(3) = 0.432$. [S]

Here's our author's solution (p. 278):

12: Since N_n, L_n, $P_{0,n}$, and P_n are all the same polynomial except possibly the form in which they are written, the error term for a Newton polynomial is the same as that for a Lagrange polynomial:

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

In this particular case, we have

$$\begin{aligned} f(x) - P_n(x) &= \frac{f^{(3)}(\xi_2)}{3!}(2-1)(2-2.2)(2-3) \\ &= \frac{1}{30}f^{(3)}(\xi_2). \end{aligned}$$

Since all derivatives are bounded between -2 and 1 over the interval $[1, 3]$, $|f^{(3)}(\xi_2)| \leq 2$ and, therefore, the error has bound

$$|f(x) - P_n(x)| \leq \frac{2}{30} = \frac{1}{15} = .0\overline{6}.$$

In[3681]:=
```
data = {{1, 0.987}, {2.2, -0.123}, {3, 0.432}};
{xpoints, ypoints} = Transpose[data];
newtonCoefs = dividedDifference[data, True]
ystar = interpolator[2, newtonCoefs, xpoints, False]
Show[
 Plot[{Abs[2 (x - 1) (x - 2.2) (x - 3) / 3 !], 2 / 30},
  {x, 1, 3}, PlotLabel → "Error bound at x=2.0"],
 ListPlot[{{2, ystar}}],
 PlotRange → All
]
```

$$\begin{pmatrix} 0.987 & -0.925 & 0.809375 \\ -0.123 & 0.69375 & 0 \\ 0.432 & 0 & 0 \end{pmatrix}$$

Out[3683]= {0.987, −0.925, 0.809375}

Out[3684]= −0.0998750000000001

Out[3685]=