CHAPTER 15

15.1 (a) Newton's polynomial. Ordering of points:

$x_1 = 3$ $f(x_1) = 6.5$
$x_2 = 4$ $f(x_2) = 2$
$x_3 = 2.5$ $f(x_3) = 7$
$x_4 = 5$ $f(x_4) = 0$

Note that based purely on the distance from the unknown, the fourth point would be (2, 5). However, because it provides better balance and is located only a little bit farther from the unknown, the point at (5, 0) is chosen.

First order:

$$f_1(3.4) = 6.5 + \frac{2 - 6.5}{4 - 3}(3.4 - 3) = 6.5 + (-4.5)(3.4 - 3) = 4.7$$

Second order:

$$f_2(3.4) = 4.7 + \frac{\frac{7 - 2}{2.5 - 4} - (-4.5)}{2.5 - 3}(3.4 - 3)(3.4 - 4)$$

$$= 4.7 + \frac{-3.333333 - (-4.5)}{2.5 - 3}(3.4 - 3)(3.4 - 4)$$

$$= 4.7 + (-2.333333)(3.4 - 3)(3.4 - 4) = 5.259887$$

Third order:

$$f_3(3.4) = 5.259887 + \frac{\frac{\frac{0 - 7}{5 - 2.5} - (-3.333333)}{5 - 4} - (-2.333333)}{5 - 3}(3.4 - 3)(3.4 - 4)(3.4 - 2.5)$$

$$= 5.259887 + \frac{\frac{-2.8 - (-3.333333)}{5 - 4} - (-2.333333)}{5 - 3}(3.4 - 3)(3.4 - 4)(3.4 - 2.5)$$

$$= 5.259887 + \frac{0.5333333 - (-2.333333)}{5 - 3}(3.4 - 3)(3.4 - 4)(3.4 - 2.5) = 4.95152$$

(b) Lagrange polynomial.

First order:

$$f_1(3.4) = \frac{3.4 - 4}{3 - 4}6.5 + \frac{3.4 - 3}{4 - 3}2 = 4.7$$

Second order:

$$f_2(3.4) = \frac{(3.4-4)(3.4-2.5)}{(3-4)(3-2.5)}6.5 + \frac{(3.4-3)(3.4-2.5)}{(4-3)(4-2.5)}2 + \frac{(3.4-3)(3.4-4)}{(2.5-3)(2.5-4)}7 = 5.259887$$

Third order:

$$f_3(3.4) = \frac{(3.4-4)(3.4-2.5)(3.4-5)}{(3-4)(3-2.5)(3-5)}6.5 + \frac{(3.4-3)(3.4-2.5)(3.4-5)}{(4-3)(4-2.5)(4-5)}2$$
$$+ \frac{(3.4-3)(3.4-4)(3.4-5)}{(2.5-3)(2.5-4)(2.5-5)}7 + \frac{(3.4-3)(3.4-4)(3.4-2.5)}{(5-3)(5-4)(5-2.5)}0 = 4.95152$$

15.2 The points can be ordered so that they are close to and centered around the unknown. A divided-difference table can then be developed as

x	*f*(*x*)	First	Second	Third	Fourth
3	5.25	7.25	2	0.25	0
5	19.75	5.25	2.75	0.25	
2	4	8	1.75		
6	36	6.25			
1	4.75				

Note that the fact that the fourth divided difference is zero means that the data was generated with a third-order polynomial.

First order:

$$f_1(4) = 5.25 + 7.25(4-3) = 12.5$$

Second order:

$$f_2(4) = 12.5 + (4-3)(4-5)2 = 10.5$$

Third order:

$$f_3(4) = 10.5 + (4-3)(4-5)(4-2)0.25 = 10$$

Fourth order:

$$f_3(4) = 10.5 + (4-3)(4-5)(4-2)(4-6)0 = 10$$

15.3 Lagrange polynomial.

First order:

$$f_1(4) = \frac{4-5}{3-5}5.25 + \frac{4-3}{5-3}19.75 = 12.5$$

Second order:

$$f_2(4) = \frac{(4-5)(4-2)}{(3-5)(3-2)}5.25 + \frac{(4-3)(4-2)}{(5-3)(5-2)}19.75 + \frac{(4-3)(4-5)}{(2-3)(2-5)}4 = 10.5$$

Third order:

$$f_3(4) = \frac{(4-5)(4-2)(4-6)}{(3-5)(3-2)(3-6)}5.25 + \frac{(4-3)(4-2)(4-6)}{(5-3)(5-2)(5-6)}19.75$$
$$+ \frac{(4-3)(4-5)(4-6)}{(2-3)(2-5)(2-6)}4 + \frac{(4-3)(4-5)(4-2)}{(6-3)(6-5)(6-2)}36 = 10$$

15.4 (a) The points can be ordered so that they are close to and centered around the unknown. A divided-difference table can then be developed as

T, °C	c = 10 g/L	first	second	third
10	10.1	-0.214	0.0026	0.000107
15	9.03	-0.227	0.003667	
5	11.3	-0.20867		
20	8.17			

Second order:

$$f_2(4) = 10.1 - 0.214(12-10) + 0.0026(12-10)(12-15) = 9.6564$$

Third order:

$$f_3(4) = 9.6564 + 0.000107(12-10)(12-15)(12-5) = 9.65192$$

(b) First, linear interpolation can be used to generate values for T = 10 and 15 at c = 15,

$$f_1(T=10, c=15) = 10.1 + \frac{8.96-10.1}{20-10}(15-10) = 9.53$$

$$f_1(T=15, c=15) = 9.03 + \frac{8.08-9.03}{20-10}(15-10) = 8.555$$

These values can then be used to determine the result at T = 12,

$$f_1(T=12, c=15) = 9.53 + \frac{8.555-9.53}{15-10}(12-10) = 9.14$$

(c) First, quadratic interpolation can be used to generate values for $T = 5$, 10 and 15 at $c = 15$,

$$f_2(T=5, c=15) = 12.8 - 0.15(15-0) + 0.0025(15-0)(15-10) = 10.7375$$

$$f_2(T=10, c=15) = 11.3 - 0.12(15-0) + 0.0003(15-0)(15-10) = 9.5225$$

$$f_2(T=15, c=15) = 10.1 - 0.107(15-0) + 0.0006(15-0)(15-10) = 8.54$$

These values can then be used to determine the result at $T = 12$,

$$f_2(T=12, c=15) = 10.7375 - 0.243(12-5) + 0.00465(12-5)(12-10) = 9.1016$$

15.5 MATLAB can be used to generate a cubic polynomial through the first 4 points in the table,

```
>> x = [1 2 3 4];
>> fx = [3.6 1.8 1.2 0.9];
>> p = polyfit(x,fx,3)
p =
   -0.1500    1.5000   -5.2500    7.5000
```

Therefore, the roots problem to be solved is

$$1.6 = -0.15x^3 + 1.5x^2 - 5.25x + 7.5$$

or

$$f(x) = -0.15x^3 + 1.5x^2 - 5.25x + 5.9 = 0$$

Bisection can be employed the root of this polynomial. Using initial guesses of $x_l = 2$ and $x_u = 3$, a value of 2.2156 is obtained with $\varepsilon_a = 0.00069\%$ after 16 iterations.

15.6 (a) Analytical:

$$0.93 = \frac{x^2}{1+x^2}$$

$$0.93 + 0.93x^2 = x^2$$

$$0.07x^2 = 0.93$$

$$x = \sqrt{\frac{0.93}{0.07}} = 3.644957$$

(b) A quadratic interpolating polynomial can be fit to the last three points using the `polyfit` function,

```
>> format long
```

```
>> x = [3 4 5];
>> y = x.^2./(1+x.^2);
>> p = polyfit(x,y,2)
p =
  -0.01040723981900   0.11402714932127   0.65158371040724
```

Thus, the best fit quadratic is

$$f_2(x) = -0.01040724x^2 + 0.11402715x + 0.6515837$$

We must therefore find the root of

$$0.93 = -0.01040724x^2 + 0.11402715x + 0.6515837$$

or

$$f(x) = -0.01040724x^2 + 0.11402715x - 0.2784163$$

The quadratic formula yields

$$x = \frac{-0.11402715 \pm \sqrt{(0.11402715)^2 - 4(-0.01040724)(-0.2784163)}}{2(0.11402715)} = \begin{matrix} 7.2835775 \\ 3.6729442 \end{matrix}$$

Thus, the estimate is 3.67294421.

(c) A cubic interpolating polynomial can be fit to the last four points using the `polyfit` function,

```
>> format long
>> x = [2 3 4 5];
>> y=x.^2./(1+x.^2)
>> p = polyfit(x,y,3)
p =
   0.00633484162896  -0.08642533936652   0.41176470588235   0.27149321266968
```

Thus, the best fit cubic is

$$f_3(x) = 0.006334842x^3 - 0.08642534x^2 + 0.4117647x + 0.2714932$$

We must therefore find the root of

$$0.93 = 0.006334842x^3 - 0.08642534x^2 + 0.4117647x + 0.2714932$$

or

$$f(x) = 0.006334842x^3 - 0.08642534x^2 + 0.4117647x - 0.6585068$$

Bisection can be employed the root of this polynomial. Using initial guesses of $x_l = 3$ and $x_u = 4$, a value of 3.61883 is obtained.

15.7 (a) Because they bracket the unknown, the two last points are used for linear interpolation,

$$f_1(0.118) = 6.5453 + \frac{6.7664 - 6.5453}{0.12547 - 0.11144}(0.118 - 0.11144) = 6.6487$$

(b) The quadratic interpolation can be implemented easily in MATLAB,

```
>> v = [0.10377 0.1144 0.12547];
>> s = [6.4147 6.5453 6.7664];
>> p = polyfit(v,s,2)
p =
  354.2358  -64.9976    9.3450
>> polyval(p,0.118)
ans =
    6.6077
```

Therefore, to the level of significance reported in the table the estimated entropy is 6.6077

(c) The inverse interpolation can be implemented in MATLAB. First, as in part **(b)**, we can fit a quadratic polynomial to the data to yield,

```
p =
  354.2358  -64.9976    9.3450
```

We must therefore find the root of

$$6.45 = 354.2358x^2 - 64.9976x + 9.3450$$

or

$$6.45 = 354.2358x^2 - 64.9976x + 2.8950$$

In MATLAB, we can generate this polynomial by subtracting 6.45 from the constant coefficient of the polynomial

```
>> p(3)=p(3)-6.45
p =
  354.2358  -64.9976    2.8950
```

Then, we can use the `roots` function to determine the solution,

```
>> roots(p)
ans =
    0.1074
    0.0761
```

Thus, the value of the specific volume corresponding to an entropy of 6.45 is 0.1074.

15.8 This problem is nicely suited for the Newton interpolating polynomial. First, we can order the data so that the points are closest to and centered around the unknown,

T	*D*
300	1.139
350	0.967
400	0.854
250	1.367
450	0.759
200	1.708

Then we can generate the divided difference table,

T	D	first	second	third	fourth	fifth
300	1.139	-0.003440	1.18000E-05	4.00000E-09	-2.93333E-10	-2.77333E-12
350	0.967	-0.002260	1.16000E-05	-4.00000E-08	-1.60000E-11	
400	0.854	-0.003420	7.60000E-06	-3.76000E-08		
250	1.367	-0.003040	1.51200E-05			
450	0.759	-0.003796				
200	1.708					

First-order (linear) fit:

$$f_1(330) = 1.139 - 0.00344(330 - 300) = 1.0358$$

Thus, the linear estimate is 1.036 to the level of significant digits provided in the original data.

Second-order (quadratic) fit:

$$f_2(330) = 1.0358 + 1.18 \times 10^{-5}(330 - 300)(330 - 350) = 1.0287$$

The quadratic estimate is 1.029 to the level of significant digits provided in the original data.

Third-order (cubic) fit:

$$f_3(330) = 1.0287 + 4 \times 10^{-9}(330 - 300)(330 - 350)(330 - 400) = 1.028888$$

The cubic estimate is also 1.029.

Fourth-order (quartic) fit:

$$f_4(330) = 1.0289 - 2.93333^{-10}(330 - 300)(330 - 350)(330 - 400)(330 - 250) = 1.0279$$

The quartic estimate now seems to be diverging slightly by moving to a value of 1.028. This may be an initial indication that the higher-order terms are beginning to induce slight oscillations.

Fifth-order (quintic) fit:

$$f_2(330) = 1.0279 - 2.77333^{-12}(330-300)(330-350)(330-400)(330-250)(330-450) = 1.02902$$

Oscillations are now evidently occurring as the fifth-order estimate now jumps back up to slightly above a value of 1.029.

On the basis of the foregoing, I would conclude that the cubic equation provides the best approximation and that a value of 1.029 is a sound estimate to the level of significant digits provided in the original data.

Inverse interpolation can be now used to determine the temperature corresponding to the value of density of 1.029. First, MATLAB can be used to fit a cubic polynomial through the four points that bracket this value. Interestingly, because of the large values of the temperatures, we get an error message,

```
>> T = [250 300 350 400];
>> D =[1.3670 1.139 0.967 0.854];
>> p = polyfit(T,D,3)
Warning: Polynomial is badly conditioned. Remove repeated data points
         or try centering and scaling as described in HELP POLYFIT.
(Type "warning off MATLAB:polyfit:RepeatedPointsOrRescale" to suppress this
warning.)
> In polyfit at 78

p =
    0.0000    0.0000   -0.0097    3.2420
```

Let's disregard this warn and proceed to adjust the polynomial so that it can be used to solve the inverse interpolation problem. To do this, we subtract the specified value of the density from the polynomial's constant coefficient

```
>> p(4)=p(4)-1.029

p =
    0.0000    0.0000   -0.0097    2.2130
```

Then we can use the `roots` function to determine the temperature that corresponds to this value

```
>> roots(p)
ans =
  1.0e+003 *
   -2.8237
    0.5938
    0.3300
```

Thus, even though the polynomial is badly conditioned one of the roots corresponds to $T = 330$ as expected.

Now let's perform the inverse interpolation, but with scaling. To do this, we will merely subtract the value at the midpoint of the temperature range (325) from all the temperatures. This acts to both reduce the magnitudes of the temperatures and centers them on zero,

```
>> format long
>> D = [1.3670 1.139 0.967 0.854];
>> T = [250 300 350 400];
>> T = T - 325;
```

Then, the cubic fit can be generated with no error message,

```
>> p = polyfit(T,D,3)
p =
   0.00000000400000   0.00001150000000  -0.00344250000000   1.04581250000000
```

We can set up the roots problem

```
>> p(4)=p(4)-1.029
p =
   0.00000000400000   0.00001150000000  -0.00344250000000   0.01681250000000
```

We can then use the `roots` function to determine the temperature that corresponds to the given density

```
>> r = roots(p)
ans =
  1.0e+003 *
  -3.14874694489127
   0.26878060289231
   0.00496634199799
```

By adding back the offset of 325, we arrive at the expected result of 330,

```
>> Tinv=r(3)+325
Tinv =
    3.299663419979927e+002
```

15.9 A MATLAB session provides a handy way to solve this problem

```
>> i = [-2 -1 -0.5 0.5 1 2];
>> V = [-637 -96.5 -20.5 20.5 96.5 637];
>> p = polyfit(i,V,5)
p =
    0.0000    0.0000   74.0000   -0.0000   22.5000   -0.0000
```

The interpolating polynomial is therefore

$$V = 74i^3 + 22.5i$$

The `polyval` function can be used to determine the interpolation at $i = 0.1$,

```
>> polyval(p,0.10)
ans =
    2.3240
```

15.10

Errata: In the first printing, the problem statement asked for a fifth-order interpolating polynomial which cannot be obtained with the 5 given points. Subsequent printings only ask for third-order and fourth-order polynomials.

Third-order case: The MATLAB polyfit function can be used to generate the cubic polynomial and perform the interpolation,

```
>> x = [1.8 2 2.2 2.4];
>> J = [0.5815 0.5767 0.5560 0.5202];
>> p = polyfit(x,J,3)
p =
    0.0167   -0.2988    0.9306   -0.2228
>> Jpred=polyval(p,2.1)
Jpred =
    0.5683
```

The built-in function `besselj` can be employed to determine the true value which can then be used to determine the percent relative error

```
>> Jtrue=besselj(1,2.1)
Jtrue =
    0.5683
>> ea=abs((Jtrue-Jpred)/Jtrue)*100
ea =
  8.1573e-004
```

Fourth-order case:

```
>> x = [1.8 2 2.2 2.4 2.6];
>> J = [0.5815 0.5767 0.5560 0.5202 0.4708];
>> p = polyfit(x,J,4)
```

```
p =
    0.0182   -0.1365    0.1818    0.2630    0.1237
>> Jpred=polyval(p,2.1)
Jpred =
    0.5683
>> ea=abs((Jtrue-Jpred)/Jtrue)*100
ea =
    0.0021
```

15.11 In the same fashion as Example 15.6, MATLAB can be used to evaluate each of the cases,

First order:

```
>> t = [1990 1980];
>> pop = [249.46 227.23];
>> ts = (t - 1955)/35;
>> p = polyfit(ts,pop,1);
>> polyval(p,(2000-1955)/35)
ans =
  271.6900
```

Second order:

```
>> t = [t 1970];
>> pop = [pop 205.05];
>> ts = (t - 1955)/35;
>> p = polyfit(ts,pop,2);
>> polyval(p,(2000-1955)/35)
ans =
  271.7400
```

Third order:

```
>> t = [t 1960];
>> pop = [pop 180.67];
>> ts = (t - 1955)/35;
>> p = polyfit(ts,pop,3);
>> polyval(p,(2000-1955)/35)
ans =
  273.9900
```

Fourth order:

```
>> t = [t 1950];
>> pop = [pop 152.27];
>> ts = (t - 1955)/35;
>> p = polyfit(ts,pop,4);
>> polyval(p,(2000-1955)/35)
ans =
  274.4200
```

Although the improvement is not great, the addition of each term causes the prediction for 2000 to increase. Thus, using higher-order approximations is moving the prediction closer to the actual value of 281.42 that occurred in 2000.

15.12 This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 20$ $f(x_0) = 8.17$
$x_1 = 15$ $f(x_1) = 9.03$
$x_2 = 25$ $f(x_2) = 7.46$
$x_3 = 10$ $f(x_3) = 10.1$
$x_4 = 30$ $f(x_4) = 6.85$
$x_5 = 5$ $f(x_5) = 11.3$
$x_6 = 0$ $f(x_6) = 12.9$

The results of applying Newton's polynomial at $T = 18$ are

Order	*f*(*x*)	Error
0	8.17000	0.344
1	8.51400	-0.018
2	8.49600	-0.00336
3	8.49264	0.00022
4	8.49286	-0.00161
5	8.49125	-0.00298
6	8.48827	

The minimum error occurs for the third-order version so we conclude that the interpolation is 8.4926.

15.13 We can use MATLAB to solve this problem,

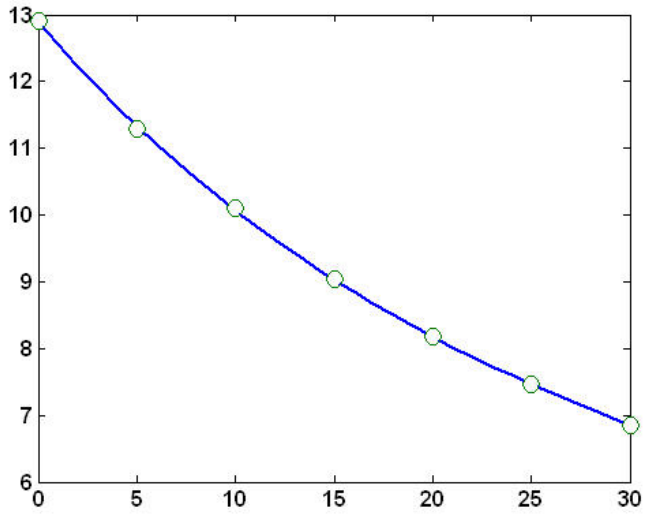
```
>> format long
>> T = [0 5 10 15 20 25 30];
>> c = [12.9 11.3 10.1 9.03 8.17 7.46 6.85];
>> p = polyfit(T,c,3)

p =
  -0.00006222222222   0.00652380952381  -0.34111111111111
12.88785714285715
```

Thus, the best-fit cubic would be

$$c = 12.8878571 - 0.34111111T + 0.00652381T^2 - 0.000062222T^3$$

We can generate a plot of the data along with the best-fit line as

```
>> Tp=[0:30];
>> cp=polyval(p,Tp);
>> plot(Tp,cp,T,c,'o')
```



We can use the best-fit equation to generate a prediction at $T = 8$ as

```
>> y=polyval(p,8)
y =
  10.54463428571429
```

15.14 This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 740$ $f(x_0) = 0.1406$
$x_1 = 760$ $f(x_1) = 0.15509$
$x_2 = 720$ $f(x_2) = 0.12184$
$x_3 = 780$ $f(x_3) = 0.16643$
$x_4 = 700$ $f(x_4) = 0.0977$

The results of applying Newton's polynomial at $T = 750$ are

Order	*f*(*x*)	Error
0	0.14060	0.007245
1	0.14785	0.000534
2	0.14838	-7E-05
3	0.14831	0.00000
4	0.14831	

Note that the third-order polynomial yields an exact result, and so we conclude that the interpolation is 0.14831.

15.15

$$m = \frac{4.6}{10} = 0.46 \qquad n = \frac{14}{10} = 1.4$$

Use the following values

$x_0 = 0.3$ $f(x_0) = 0.08561$
$x_1 = 0.4$ $f(x_1) = 0.10941$
$x_2 = 0.5$ $f(x_2) = 0.13003$
$x_3 = 0.6$ $f(x_3) = 0.14749$

MATLAB can be used to perform the interpolation

```
>> format long
>> x=[0.3 0.4 0.5 0.6];
>> y=[0.08561 0.10941 0.13003 0.14749];
>> a=polyfit(x,y,3)

a =
  0.00333333333331  -0.16299999999997   0.35086666666665  -0.00507000000000
>> polyval(a,0.46)
ans =
  0.12216232000000
```

This result can be used to compute the vertical stress

$$q = \frac{100}{4.6(14)} = 1.552795$$

$$\sigma_z = 1.552795(0.1221623) = 0.189693$$

15.16 Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 1.25$ $f(x_0) = 0.7$
$x_1 = 0.75$ $f(x_1) = -0.6$
$x_2 = 1.5$ $f(x_2) = 1.88$
$x_3 = 0.25$ $f(x_3) = -0.45$
$x_4 = 2$ $f(x_4) = 6$

The results of applying Newton's polynomial at $i = 1.15$ are

Order	*f*(*x*)	Error
0	0.7	-0.26
1	0.44	-0.11307
2	0.326933	-0.00082
3	0.326112	0.011174
4	0.337286	

The minimum error occurs for the second-order version so we conclude that the interpolation is 0.3269.

15.17 Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

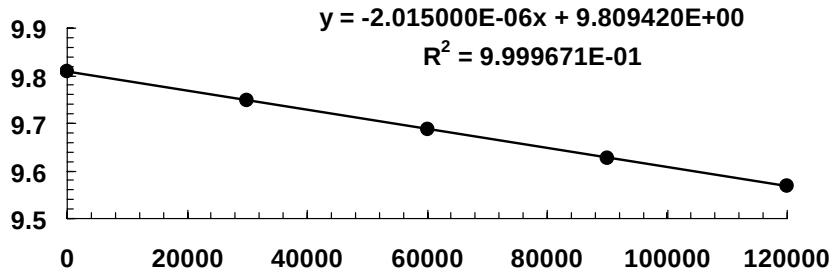
$x_0 = 0.25$ $f(x_0) = 7.75$
$x_1 = 0.125$ $f(x_1) = 6.24$
$x_2 = 0.375$ $f(x_2) = 4.85$
$x_3 = 0$ $f(x_3) = 0$
$x_4 = 0.5$ $f(x_4) = 0$

The results of applying Newton's polynomial at $t = 0.23$ are

Order	*f*(*x*)	Error
0	7.75	-0.2416
1	7.5084	0.296352
2	7.804752	0.008315
3	7.813067	0.025579
4	7.838646	

The minimum error occurs for the second-order version so we conclude that the interpolation is 7.805.

15.18 Using linear regression gives



A prediction of g at 55,000 m can be made as

$$g(55{,}000) = -2.015 \times 10^{-6}(55{,}000) + 9.80942 = 9.6986$$

Note that we can also use linear interpolation to give

$$g_1(55{,}000) = 9.698033$$

Quadratic interpolation yields

$$g_2(55{,}000) = 9.697985$$

Cubic interpolation gives

$g_3(55{,}000) = 9.69799$

Based on all these estimates, we can be confident that the result to 3 significant digits is 9.698.

15.19 This part of the problem is well-suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible, for $x = 4$.

$y_0 = 4$ $f(y_0) = 38.43$
$y_1 = 2$ $f(y_1) = 53.5$
$y_2 = 6$ $f(y_2) = 30.39$
$y_3 = 0$ $f(y_3) = 80$
$y_4 = 8$ $f(y_4) = 30$

The results of applying Newton's polynomial at $y = 3.2$ are

Order	*f(y)*	Error
0	38.43	6.028
1	44.458	-0.8436
2	43.6144	-0.2464
3	43.368	0.112448
4	43.48045	

The minimum error occurs for the third-order version so we conclude that the interpolation is 43.368.

(b) This is an example of two-dimensional interpolation. One way to approach it is to use cubic interpolation along the y dimension for values at specific values of x that bracket the unknown. For example, we can utilize the following points at $x = 2$.

$y_0 = 0$ $f(y_0) = 90$
$y_1 = 2$ $f(y_1) = 64.49$
$y_2 = 4$ $f(y_2) = 48.9$
$y_3 = 6$ $f(y_3) = 38.78$

$T(x = 2, y = 2.7) = 58.13288438$

All the values can be tabulated as

$T(x = 2, y = 2.7) = 58.13288438$
$T(x = 4, y = 2.7) = 47.1505625$
$T(x = 6, y = 2.7) = 42.74770188$
$T(x = 8, y = 2.7) = 46.5$

These values can then be used to interpolate at $x = 4.3$ to yield

$T(x = 4.3, y = 2.7) = 46.03218664$

Note that some software packages allow you to perform such multi-dimensional interpolations very efficiently. For example, MATLAB has a function interp2 that provides numerous options

for how the interpolation is implemented. Here is an example of how it can be implemented using linear interpolation,

```
>> Z=[100 90 80 70 60;
85 64.49 53.5 48.15 50;
70 48.9 38.43 35.03 40;
55 38.78 30.39 27.07 30;
40 35 30 25 20];
>> X=[0 2 4 6 8];
>> Y=[0 2 4 6 8];
>> T=interp2(X,Y,Z,4.3,2.7)

T =
   47.5254
```

It can also perform the same interpolation but using bicubic interpolation,

```
>> T=interp2(X,Y,Z,4.3,2.7,'cubic')
T =
   46.0062
```

Finally, the interpolation can be implemented using splines,

```
>> T=interp2(X,Y,Z,4.3,2.7,'spline')
T =
   46.1507
```

15.20 (a) Linear interpolation:

$$s = 6.4147 + \frac{6.5453 - 6.4147}{0.11144 - 0.10377}(0.108 - 0.10377) = 6.486726$$

(b) Quadratic interpolation:

$$s = 6.486726 + \frac{15.83811 - 17.02738}{0.1254 - 0.10377}(0.108 - 0.10377)(0.108 - 0.11144) = 6.487526$$

(c) Inverse interpolation. First, we fit a quadratic to the data

$$f_2(v) = 4.011945 + 28.860154v - 54.982456v^2$$

The roots problem can then be developed by setting this polynomial equal to the desired value of 6.6 to give

$$f_2(v) = -2.588055 + 28.860154v - 54.982456v^2$$

The quadratic formula can then be used to determine the root as

$$v = \frac{-28.860154 + \sqrt{(28.860154)^2 - 4(-54.982456)(-2.588055)}}{2(-54.982456)} = 0.11477$$