Chapter 3: Solving Algebraic Equations

Solve

Given any algebraic equation, *Mathematica* can solve for the variable. We define a function called *eqn1* and we want the value of *x* when *eqn1* equals 4. Note the use of the double equals sign ==. This symbolizes an equality, while a single equals sign stands for an assignment.

```
eqn1 = 2 x + 3;
Solve[eqn1 == 4, x]
```

$$\left\{\left\{x \to \frac{1}{2}\right\}\right\}$$

Sometimes you need to solve for more than one variable. Remember that you need at least one equation per variable. We define a second function called *eqn2* and set that equal to *7* to find *x* and *y*.

```
eqn2 = x + 3 y;
Solve[{eqn1 == 4, eqn2 == 7}, {x, y}]
```

$$\left\{\left\{x \to \frac{1}{2},\ y \to \frac{13}{6}\right\}\right\}$$

Solve can also be used to find the roots of a polynomial:

```
poly1 = x^3 + 3 x^2 + 2;
Solve[poly1 == 0, x]
```

$$\left\{\left\{x \to -1 - \frac{1}{\left(2-\sqrt{3}\right)^{1/3}} - \left(2-\sqrt{3}\right)^{1/3}\right\},\right.$$

$$\left\{x \to -1 + \frac{1}{2}\left(2-\sqrt{3}\right)^{1/3}\left(1-i\sqrt{3}\right) + \frac{1+i\sqrt{3}}{2\left(2-\sqrt{3}\right)^{1/3}}\right\},$$

$$\left.\left\{x \to -1 + \frac{1-i\sqrt{3}}{2\left(2-\sqrt{3}\right)^{1/3}} + \frac{1}{2}\left(2-\sqrt{3}\right)^{1/3}\left(1+i\sqrt{3}\right)\right\}\right\}$$

Flatten and First

The problem with *Solve* is that it'll give you ALL the possible solutions including the imaginary ones. The answers are always given in a list, which can be difficult to read. Use *Flatten* to remove the list brackets.

```
soln1 = Flatten[Solve[poly1 == 0, x]]
```

$$\left\{x \to -1 - \frac{1}{\left(2-\sqrt{3}\right)^{1/3}} - \left(2-\sqrt{3}\right)^{1/3},\right.$$

$$x \to -1 + \frac{1}{2}\left(2-\sqrt{3}\right)^{1/3}\left(1 - i\sqrt{3}\right) + \frac{1 + i\sqrt{3}}{2\left(2-\sqrt{3}\right)^{1/3}},$$

$$\left.x \to -1 + \frac{1 - i\sqrt{3}}{2\left(2-\sqrt{3}\right)^{1/3}} + \frac{1}{2}\left(2-\sqrt{3}\right)^{1/3}\left(1 + i\sqrt{3}\right)\right\}$$

If the first solution is the one you want, you can use *First* to display only the first solution of your list. Make sure to look at all the solutions before using this command.

```
soln2 = First[Solve[poly1 == 0, x]]
```

$$\left\{x \to -1 - \frac{1}{\left(2-\sqrt{3}\right)^{1/3}} - \left(2-\sqrt{3}\right)^{1/3}\right\}$$

A better way to use these commands is to put them after your input using double slanted bars.

```
Solve[poly1 == 0, x] // Flatten // First
```

$$x \to -1 - \frac{1}{\left(2-\sqrt{3}\right)^{1/3}} - \left(2-\sqrt{3}\right)^{1/3}$$

NSolve

The first solution to *eqn1* and *eqn2* was given symbolically. *Mathematica* will not give decimal answers without the *N* command. For a numerical approximation of an output using *Solve*, the command is *NSolve*.

```
NSolve[poly1 == 0, x] // Flatten // First
```

$x \to -3.19582$

Another way to do this is to put the *N* command at the end of the output:

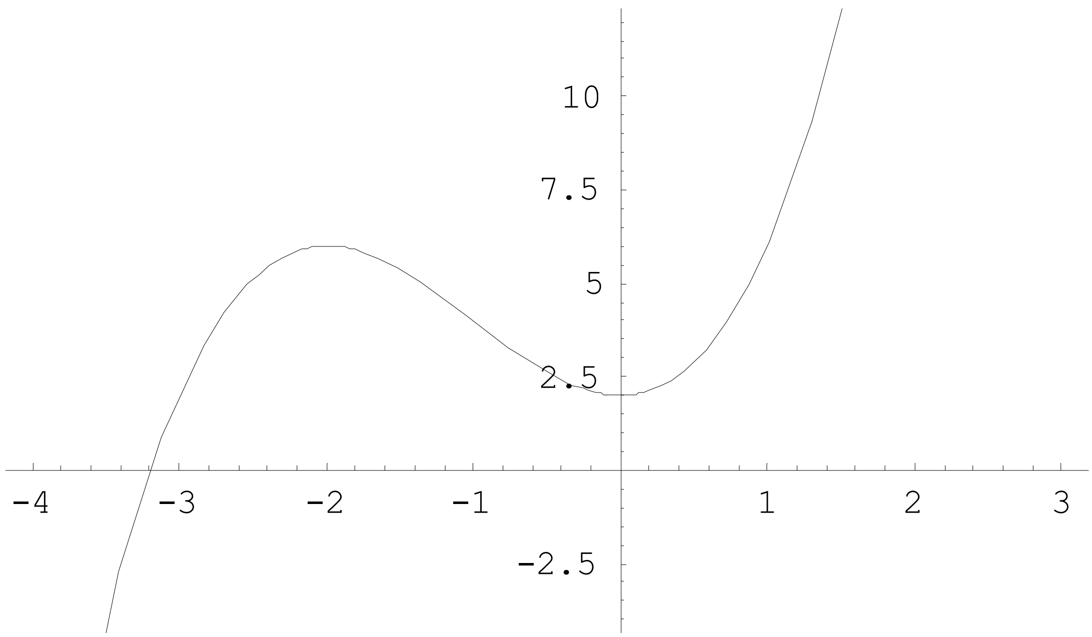
```
Solve[poly1 == 0, x] // Flatten // First // N
```

$x \to -3.19582$

FindRoot

Unlike *Solve*, *FindRoot* can only give one answer at a time. The answer that it gives is based on your best guess as to the location of the root. This is best done using a plot where you know the general value of the root. Using *Plot*, graph *poly1* and adjust the range of *x* to get a good approximation of the root.

```
Plot[poly1, {x, -4, 3}];
```



The root looks to be close to $x = -3$. *FindRoot* requires three arguments: the function, the variable, and the guess. The last two arguments must be in a list.

```
rootsoln = FindRoot[poly1 == 0, {x, -3}]
```

{x → −3.19582 }

```
Print[rootsoln, "is a solution to ", poly1, " = 0"]
```

{x → −3.19582 }is a solution to $2 + 3x^2 + x^3$ = 0

From Physical Chemistry, 6th Edition by Peter Atkins:

Exercise 1.12
The density of air at various temperatures is given below:

ρ, g/L	**1.877**	**1.294**	**0.946**
T, °C	**-85**	**0**	**100**

Using Charles' Law, determine a value for the temperature, in °C at absolute zero.

Absolute zero is also known as 0 K.

Volume = 0 L at absolute zero by definition.

Assume a 1.000g sample of air and calculate the volume at each temperature using density.

Make a list of volume and temperature values and transpose to get a list of {T, V} points.

Use *ListPlot* to scatter plot the points and fit an equation using *Fit*.

From the equation, use *FindRoot* or *Solve* to determine the root of *T*, when *V = 0*.

Finally, print out the answer with units of °C.

Density: $\rho = \dfrac{\text{mass}}{\text{volume}}$

Charles' Law: $\dfrac{V_1}{T_1} = \dfrac{V_2}{T_2}$

Answer: Exercise 1.12

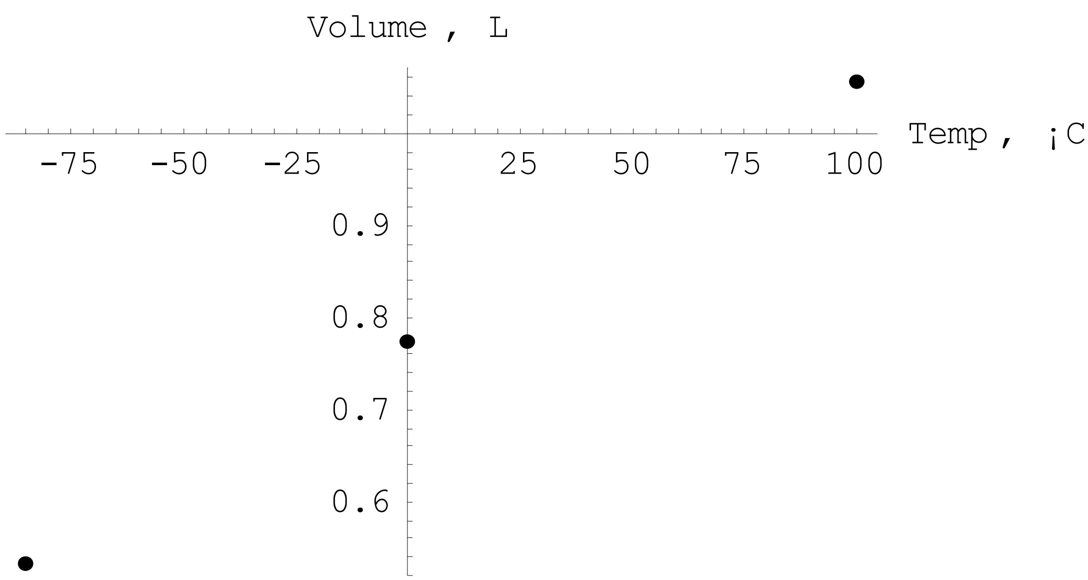
Make a list of densities and a list of temperatures:

```
ρ = {1.877, 1.294, 0.946};
tempC = {-85, 0, 100};
vol = 1.000 / ρ;
points = {tempC, vol}
{{-85, 0, 100}, {0.532765 , 0.772798 , 1.05708 }}
```

```
data = Transpose[points]
{{-85, 0.532765 }, {0, 0.772798 }, {100 , 1.05708 }}
```

Make a plot of the data list:

```
plot1 = ListPlot[data, PlotStyle → PointSize[0.02],
   AxesLabel → {"Temp, °C", "Volume, L"}];
```

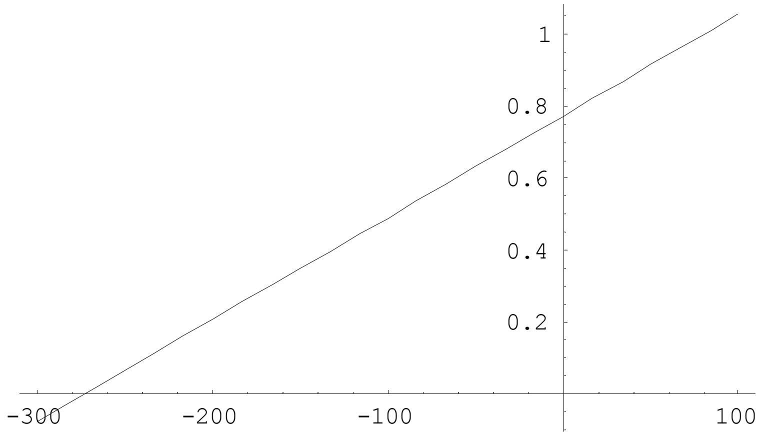


Fit an equation to the points using *Fit*:

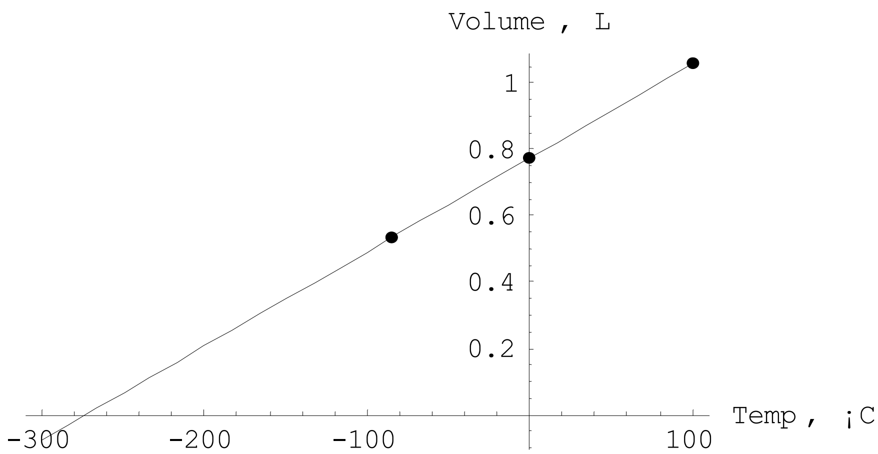
```
fiteqn = Fit[data, {1, x}, x]
0.773376 + 0.0028344 x
```

Plot the equation and adjust the range until the line crosses the x-axis:

```
fitplot = Plot[fiteqn, {x, -300, 100}];
```




```
Show[{plot1, fitplot}];
```



Find the absolute zero temperature at 0 L using *FindRoot*:

```
abszero = FindRoot[fiteqn == 0, {x, -300}]
{x → -272.853 }

Print["The temperature at absolute zero is ",
 abszero, " °C"]
The temperature  at absolute  zero  is {x → -272.853 }  °C
```

Or using *Solve*:

```
abszero = Solve[fiteqn == 0, x] // Flatten
{x → -272.853 }

Print["The temperature at absolute zero is ", abszero,
 " °C"]
The temperature  at absolute  zero  is {x → -272.853 }  °C
```