Numerical methods

Henrik Kenneth Andersen

2021-10-08

Say we had the function

$$f(x) = 2x^2 + 6x - 8. \tag{1}$$

which is shown in Figure 1.

```
library(latex2exp)

fx <- function(x) {
  y <- 2 * x^2 + 6 * x - 8
  return(y)
}

df <- data.frame(x = seq(-6, 4, 0.1), y = fx(seq(-6, 4, 0.1)))

library(ggplot2)

ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$")) +
  geom_hline(yintercept = 0)
```

If we wanted to find the root of the function, i.e., where the function equals zero, we could solve for $f(x) = 0$

$$\begin{aligned} f(x) &= 2x^2 + 6x - 8 \\ 0 &= 2x^2 + 6x - 8 \\ 0/2 &= x^2 + 3x - 4 \end{aligned} \tag{2}$$

which is in the 'monic form' $x^2 + px + q$, where $p = 3$, $q = -4$. We can solve this with the quadratic formula

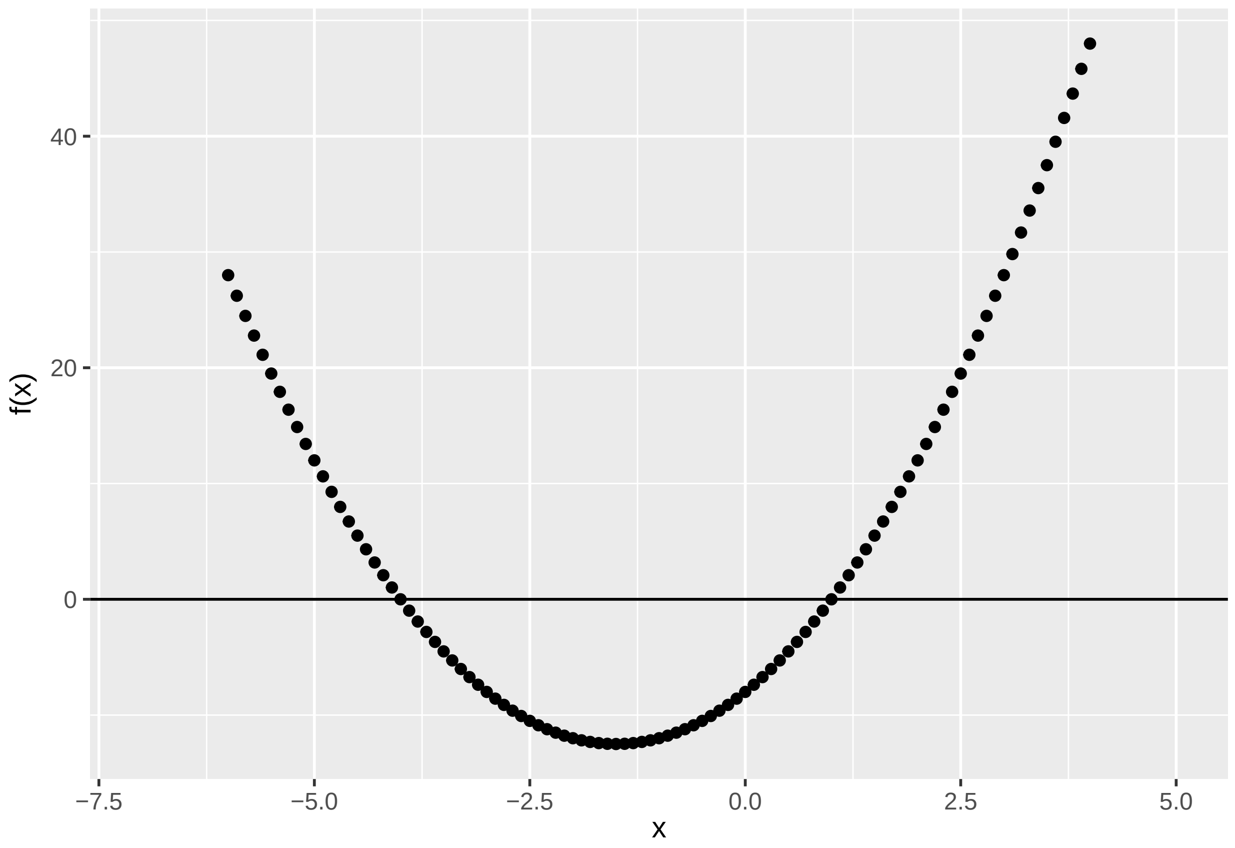

Figure 1: $f(x) = 2x^2 + 6x - 8$

$$
\begin{aligned}
x &= \frac{1}{2}(-p \pm \sqrt{p^2 - 4q}) \\
&= \frac{1}{2}(-3 \pm \sqrt{3^2 - 4(-4)}) \\
&= \frac{1}{2}(-3 \pm \sqrt{25}) \\
&= \frac{1}{2}(-3 + 5) = \frac{1}{2}2 = 1, \\
&= \frac{1}{2}(-3 - 5) = \frac{1}{2} - 8 = -4.
\end{aligned} \tag{3}
$$

The points where $f(x) = 0$ are -4 and 1 which is clear from the graph.

Another way to find the roots of the function is using Newton's method. We start at any value for x and take the first derivative of the function. This gives us the tangent line for that value of x.

The first derivative is

$$f(x) = 2x^2 + 6x - 8 \tag{4}$$
$$f'(x) = 4x + 6 \tag{5}$$

The slope at, say, $x = -6$ is $4(-6) + 6 = -18$.

```r
fpx <- function(x) {
  y <- 4 * x + 6
}

ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_abline(intercept = -fpx(-6) * -6 + fx(-6), slope = fpx(-6), linetype = "dashed") +
  geom_vline(xintercept = -6, linetype = "dashed") +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$")) +
  annotate("text", x = -5.05, y = -5, label = TeX("$x_{0} - x_{1}$")) +
  annotate("text", x = -6.50, y = 10, label = TeX("f(x)")) +
  annotate("text", x = -4.50, y = 12.5, label = TeX("f'(x)"))
```

Let is call this value the *starting value*, i.e., $x_0 = -6$.

From this, we can see that the tangent line forms a right-angle triangle with $f(x_0)$ and the distance between the starting value, -6 and the value where the tangent crosses $f(x_0) = 0$. We call the value where the tangent line crosses $f(x) = 0$ "x_1", which will be our next value in the iterative procedure.

Notice that $f'(x_0)$ is the slope, $f(x_0)$ is the rise and $x_0 - x_1$ is the run. We know that slope is "rise over run", so

$$f'(x_0) = \frac{f(x_0)}{x_0 - x_1} \tag{6}$$

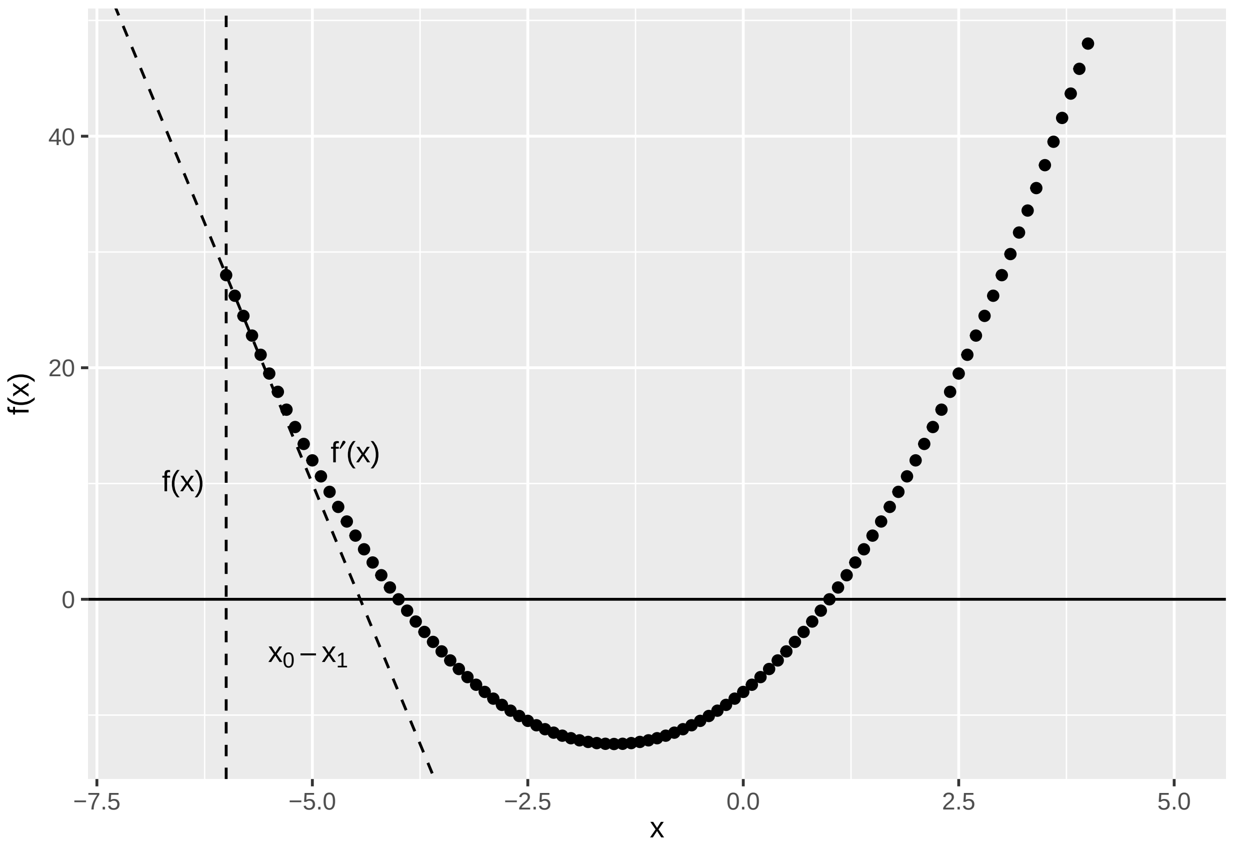

Figure 2: $f(x) = 2x^2 + 6x - 8$

and by rearrangement

$$x_0 - x_1 = \frac{f(x_0)}{f'(x_0)} \tag{7}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{8}$$

which is the formula for the iterative procedure. We take the current value for $x$ and subtract the value of $f(x)$ divided by $f'(x)$. This will give us the next step. Once we have it, we plug it back in to get the step after that, and so on.

```r
newton <- function(x) {
  xnew <- x - fx(x) / fpx(x)
  return(xnew)
}

newton(-6)
```

```
## [1] -4.444444
```

The new value is -4.44 repeating.

```r
ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_abline(intercept = -fpx(-6) * -6 + fx(-6), slope = fpx(-6),
              linetype = "dashed", color = "blue") +
  geom_abline(intercept = -fpx(-4.444) * -4.444 + fx(-4.444), slope = fpx(-4.444),
              linetype = "dashed", color = "red") +
  geom_vline(xintercept = -6, linetype = "dashed") +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$"))
```

The new value is already incredibly close to the root of the function -4. If we continue in this fashion, the new values *converge* to the analytical solution.

```r
newton(newton(-6))
```

```
## [1] -4.033543
```

```r
newton(newton(newton(-6)))
```

```
## [1] -4.000222
```

```r
newton(newton(newton(newton(-6))))
```

```
## [1] -4
```

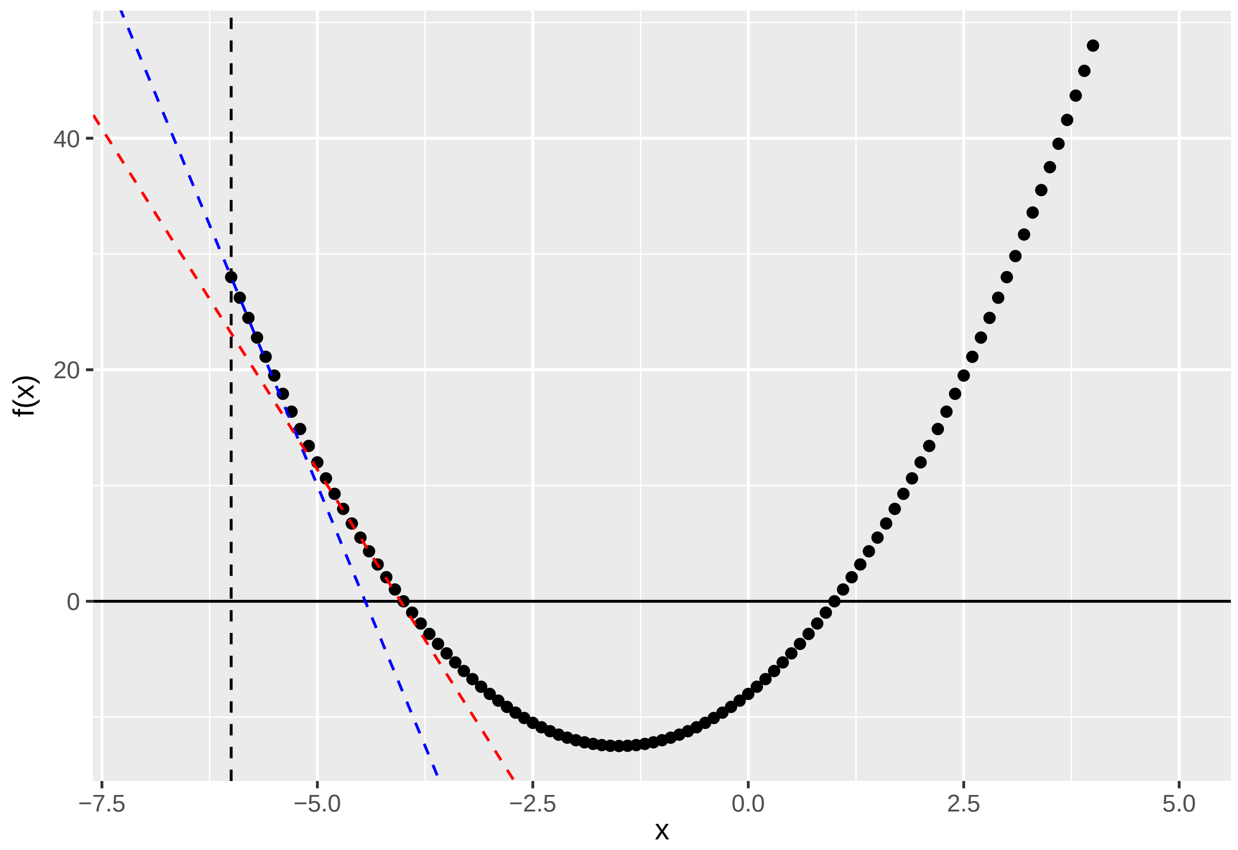

Figure 3: $f(x) = 2x^2 + 6x - 8$

```r
ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_abline(intercept = -fpx(-6) * -6 + fx(-6), slope = fpx(-6),
              linetype = "dashed", color = "blue") +
  geom_abline(intercept = -fpx(-4.444) * -4.444 + fx(-4.444), slope = fpx(-4.444),
              linetype = "dashed", color = "red") +
  geom_abline(intercept = -fpx(-4.034) * -4.034 + fx(-4.034), slope = fpx(-4.034),
              linetype = "dashed", color = "pink") +
  geom_abline(intercept = -fpx(-4.000) * -4.000 + fx(-4.000), slope = fpx(-4.000),
              linetype = "dashed", color = "purple") +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$"))
```

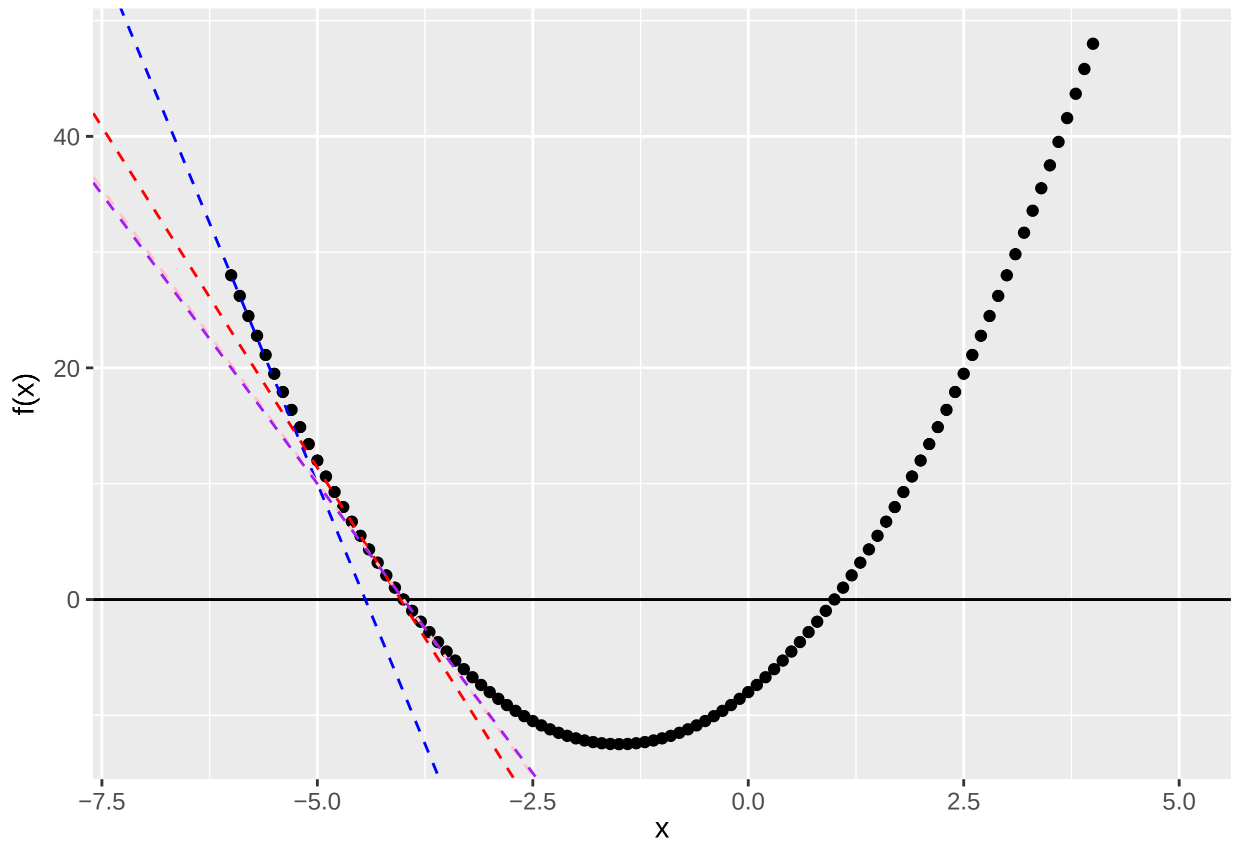

Figure 4: $f(x) = 2x^2 + 6x - 8$

After the fourth iteration, there is no noticable improvement. The slopes are lying on top of one another and the new values differ only after many decimal places.

We can also begin on the other side of the function and find the second root:

```r
newton(5)
```

```
## [1] 2.230769
```

```r
newton(newton(5))
```

```
## [1] 1.203013
```

```r
newton(newton(newton(5)))
```

```
## [1] 1.007624
```

```r
newton(newton(newton(newton(5))))
```

```
## [1] 1.000012
```

The previous sections demonstrated the use of Newton's method for finding the root of a function, where $f(x) = 0$.

But what if we want to find the minimum of this function? Then we need to find where $f'(x) = 0$, i.e., where the slope of the tangent line equals zero: a horizontal line.

Analytically, this is easy to find:

$$\begin{aligned} f'(x) &= 4x + 6 && (9)\\ 0 &= 4x + 6 && (10)\\ 4x &= -6 && (11)\\ x &= -1.5 && (12) \end{aligned}$$

```r
ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_hline(yintercept = fx(-1.5), linetype = "dashed") +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$"))
```

But we can use Newton's method here, as well. Instead of finding where $f(x) = 0$, we just need to find where $f'(x) = 0$, i.e., where the slope of the tangent line equals zero.

The analog to the solution above,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{13}$$

is to simply replace the function with the first derivative and the first derivative with the second derivative:

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \tag{14}$$

where $f''(x) = 4$. The Newton method finds the minimum in this case immediately:

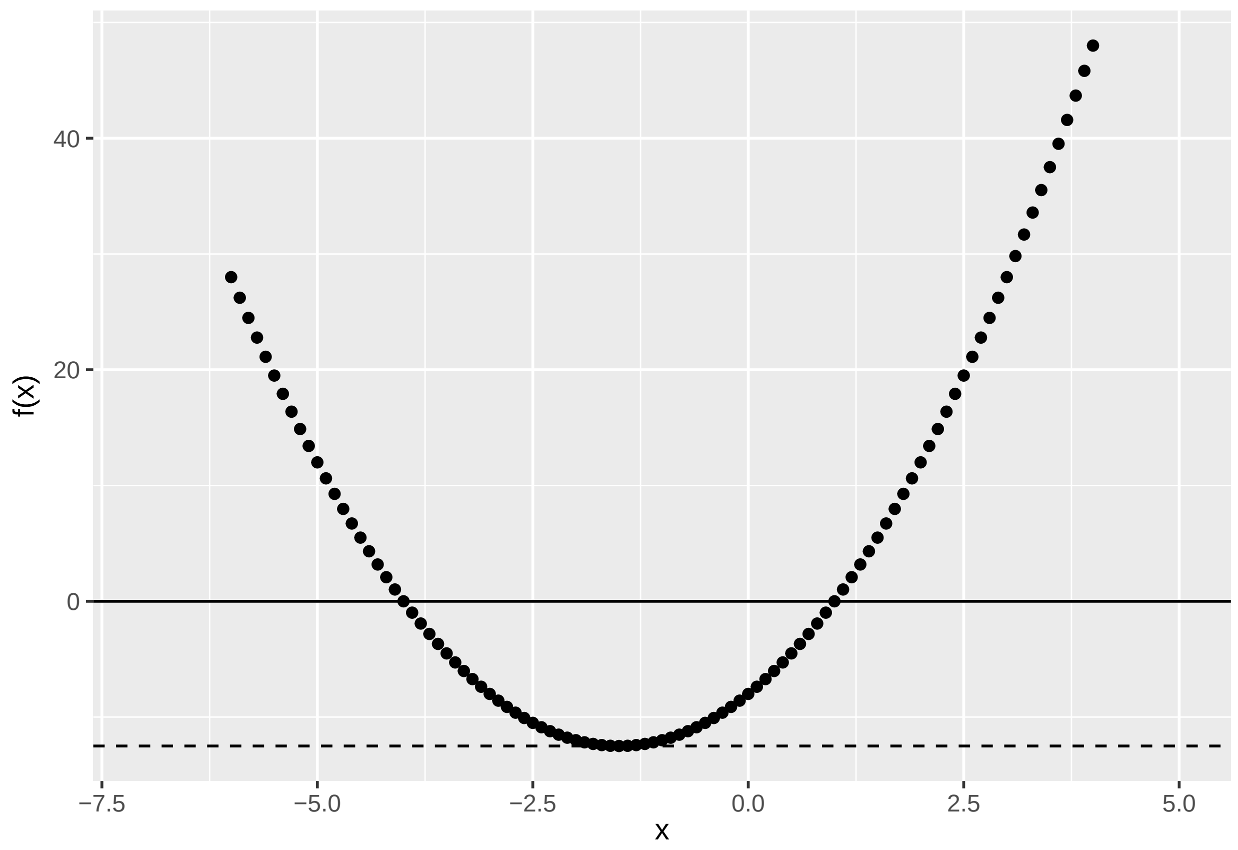

Figure 5: $f(x) = 2x^2 + 6x - 8$

```r
fppx <- 4

newton_min <- function(x) {
  xnew <- x - (fpx(x) / fppx)
  return(xnew)
}

newton_min(-6)
```

```
## [1] -1.5
```

This is because when we plot the values of x against the first partial derivative, we see a straight line:

```r
df2 <- data.frame(x = seq(-6, 4, 0.1), y = fpx(seq(-6, 4, 0.1)))

ggplot(df2, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = -1.5, linetype = "dashed")
```

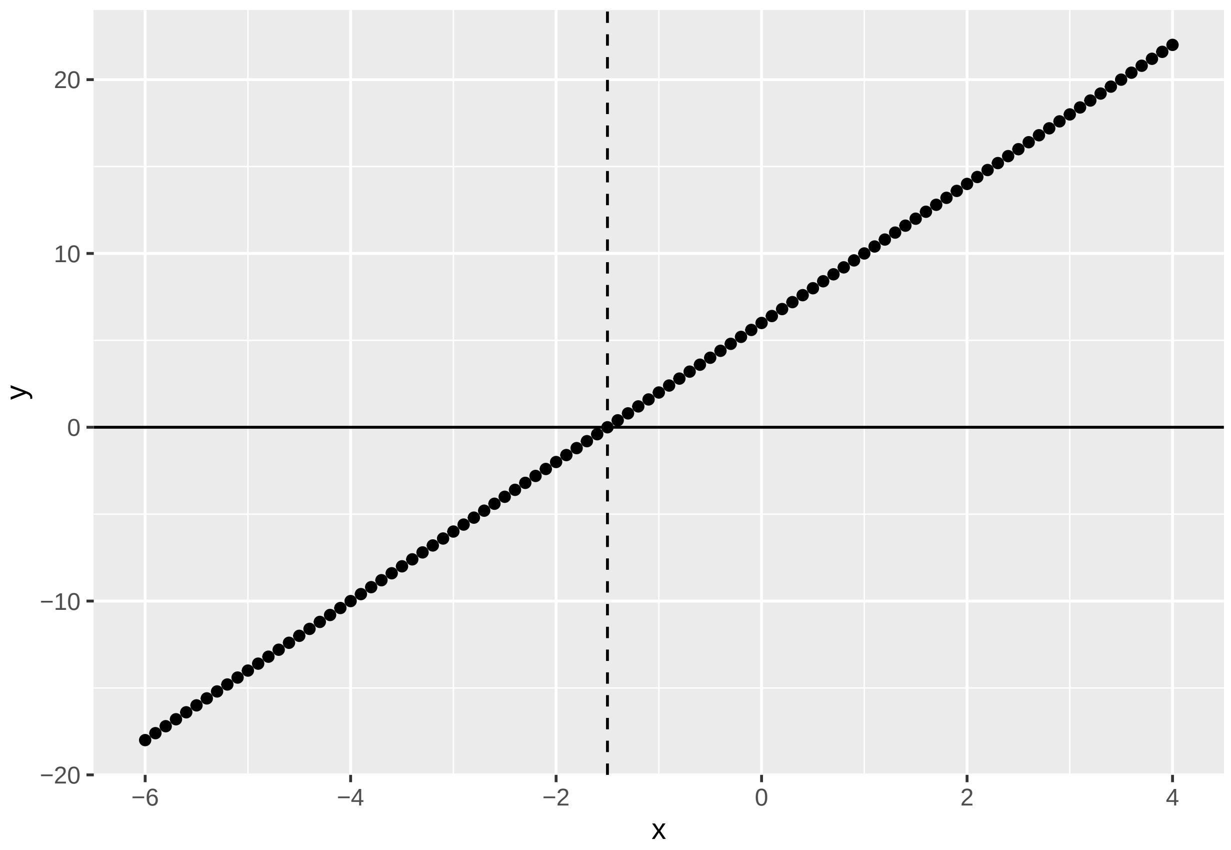

Figure 6: $f'(x) = 4x + 6$

We see that the value for x where the function ($f'(x)$, not $f(x)$) is zero is -1.5. This is a constant, so it does not matter what value we enter in `newton_min()`, the result is always the solution. The value for x that minimizes the solution is -1.5.

Another method similar to Newton's is called Gradient descent.

We take some starting value and evaluate the first derivative at that value. So far this does not differ from Newton's method.

```r
ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_abline(intercept = -fpx(-6) * -6 + fx(-6), slope = fpx(-6), linetype = "dashed") +
  geom_vline(xintercept = -6, linetype = "dashed") +
  scale_x_continuous(name = TeX("$x$"), limits = c(-7, 5)) +
  scale_y_continuous(name = TeX("$f(x)$"))
```

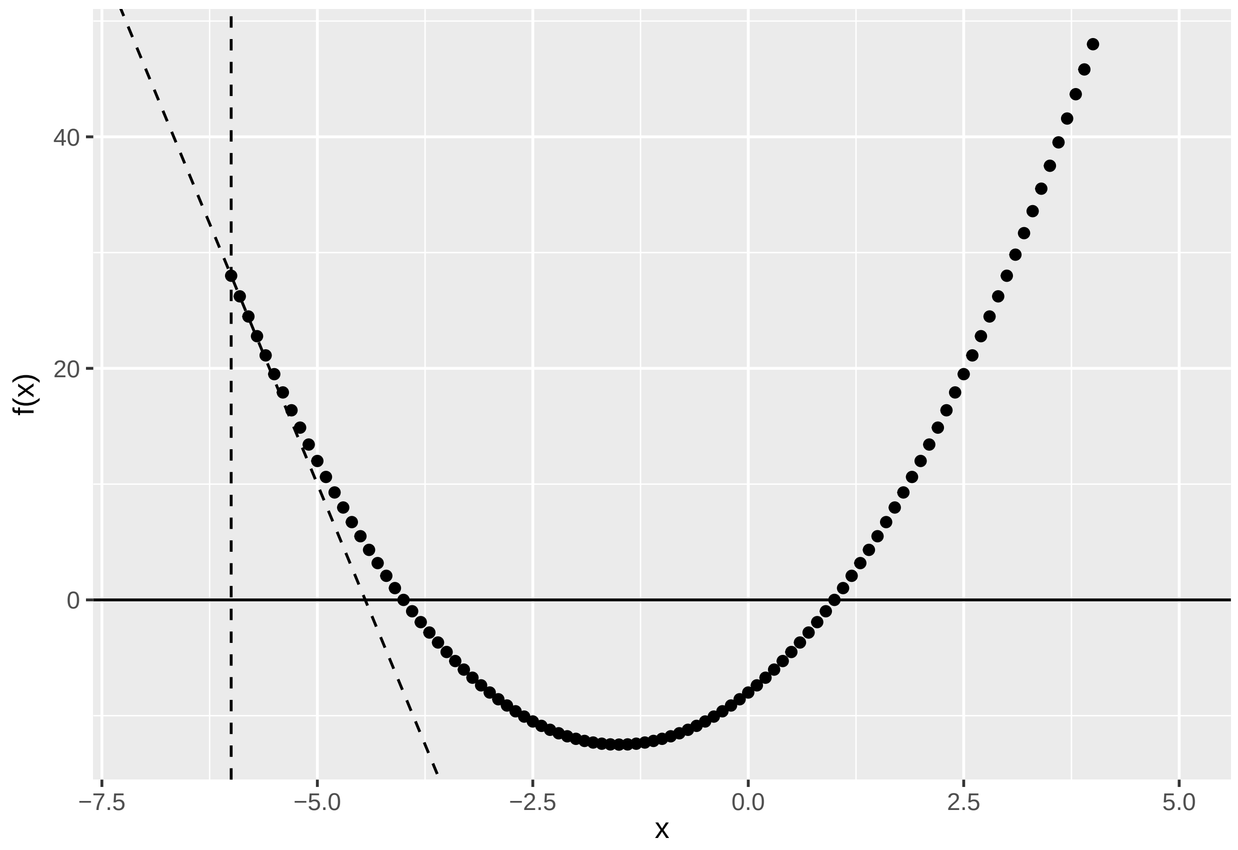

Figure 7: $f(x) = 2x^2 + 6x - 8$

Next we define an arbitrary *step value*. In Newton's method the step value was the second derivative. In the last example it was 4, so we were essentially multiplying the derivative by $\frac{1}{4}$ each time and subtracting that off of the previous value to get the next value. In gradient descent, we do not use the second derivative, but just choose some small value so that we do not 'overshoot' the minimum.

For example, let us use $\frac{1}{10} = 0.1$ as a step value.

```r
descent <- function(x) {
  xnew <- x - fpx(x) * 0.1
  return(xnew)
}

descent(-6)
```

```
## [1] -4.2
```

The algorithm depends on the step value, whereas Newton's method finds the optimal step value by taking the second derivative. Here, we do not need to take the second derivative (which can be computationally difficult in some models), but the procedure is much less efficient. After 8 iterations the values are beginning to approach the 'correct' value of -1.5, but they are still relatively far off.

```r
descent(descent(-6))
```

```
## [1] -3.12
```

```r
descent(descent(descent(-6)))
```

```
## [1] -2.472
```

```r
descent(descent(descent(descent(-6))))
```

```
## [1] -2.0832
```

```r
descent(descent(descent(descent(descent(-6)))))
```

```
## [1] -1.84992
```

```r
descent(descent(descent(descent(descent(descent(-6))))))
```

```
## [1] -1.709952
```

```r
descent(descent(descent(descent(descent(descent(descent(-6)))))))
```

```
## [1] -1.625971
```

```r
descent(descent(descent(descent(descent(descent(descent(descent(-6))))))))
```

```
## [1] -1.575583
```

```r
descent(descent(descent(descent(descent(descent(descent(descent(descent(-6)))))))))
```

```
## [1] -1.54535
```

We can choose a larger step to get there quicker:

```r
descent <- function(x) {
  xnew <- x - fpx(x) * 0.2
  return(xnew)
}

descent(-6)
```

```
## [1] -2.4
```

```r
descent(descent(-6))
```

```
## [1] -1.68
```

```r
descent(descent(descent(-6)))
```

```
## [1] -1.536
```

```r
descent(descent(descent(descent(-6))))
```

```
## [1] -1.5072
```

```r
descent(descent(descent(descent(descent(-6)))))
```

```
## [1] -1.50144
```

but again, Newton's method chooses the optimal steps automatically.