Root Finding Methods

Stats 102A

Miles Chen

Department of Statistics

Week 7 Wednesday

UCLA

Section 1

Introduction - Numeric Methods

Introduction

A **numerical method** is a technique or algorithm that approximates the solution to a mathematical problem. Numerical methods are typically iterative algorithms that are implemented using computers.

Why use approximate methods?

- The mathematical problem may not have an exact (or **analytical**) solution using algebra or calculus.
- Even if an exact solution exists in theory, it may be computationally **intractable**, i.e., it would take an unreasonably large amount of resources (usually time) to compute.

Numerical methods take advantage of a computer's ability to follow instructions quickly to compute a arbitrarily good approximations to solutions that might otherwise be impossible or infeasible to compute exactly.

Introduction

As an example, consider the probability density function $f(x)$ for the **standard normal distribution** $\mathcal{N}(0,1)$, defined by

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

The cumulative distribution function for $Z \sim \mathcal{N}(0,1)$ is defined as

$$\Phi(z) = P(Z \le z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,\mathrm{d}x.$$

There is no closed form solution for $\Phi(z)$.

For any arbitrary finite value of z, the value of $\Phi(z)$ must be approximated using numerical methods. We will not cover this specific numerical method in our class.

See: Cody, W. D. (1993) Algorithm 715: SPECFUN – A portable FORTRAN package of special function routines and test drivers. ACM Transactions on Mathematical Software 19, 22–32.

Root Finding Methods

The first class of numerical methods we will discuss are **root finding** methods, which approximate the roots (or zeroes) of a function.

Suppose we have a continuous function f that accepts a real-valued input and returns a real-valued output. That is $f : \mathbb{R} \to \mathbb{R}$. A **root** of f is a solution to the equation

$$f(x) = 0$$

Graphically, a root is where the graph $y = f(x)$ crosses the x-axis.

The solution to many mathematical and statistical problems can be expressed in terms of the root of a suitable function, so these methods have many applications.

Quadratic Polynomials

You are probably already familiar with finding the roots of a quadratic polynomial.

For example, if f is a quadratic (i.e., degree 2) polynomial of the form

$$f(x) = ax^2 + bx + c,$$

for $a, b, c \in \mathbb{R}$ and $a \neq 0$, then the roots of f are given by the **quadratic formula**

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Before we go on

If an analytic solution exists and is easily found, you should use the analytic solution.

In cases where the analytic solution is difficult to obtain, you may consider using a numeric method.

References and Resources

The main reference for these notes

- Jones, O., Maillardet, R., and Robinson, A.'s "Introduction to Scientific Programming and Simulation Using R":
  https://doi.org/10.1201/9781420068740

Additional resources

- Jeffrey R. Chasnov's lecture notes on numerical methods:
  https://www.math.ust.hk/~machas/numerical-methods.pdf
- Richard Anstee's notes on the Newton-Raphson method:
  https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

Section 2

Fixed Point Iteration

Fixed Points

Fixed point iteration

Let $g : \mathbb{R} \to \mathbb{R}$ be a continuous function. A **fixed point** of g is a solution to the equation

$$g(x) = x$$

Graphically, a fixed point is where the graph $y = g(x)$ crosses the line $y = x$.

An interesting/useful property of fixed points is that they remain fixed after iteratively applying the function an arbitrary number of times. That is, if x is a fixed point of g, then

$$x = g(x) = g(g(x)) = g(g(g(x))) = \cdots$$

Attractive Fixed Points

The fixed point iteration method takes advantage of **attractive** fixed points.

A fixed point a of g is **attractive** (or **attracting**) if, for any initial value x that is "close enough" to a, the iterated function sequence

$$x, g(x), g(g(x)), g(g(g(x))), \ldots$$

converges to a.

Side Note (won't be on exam): A more formal definition is as follows. Let $g^n(x)$ denote the iterated function applying g n times. A fixed point a is attractive if there is an interval I around a such that

- If $x \in I$, then $g^n(x) \in I$, for all $n > 0$.
- For all $x \in I$, $g^n(x) \to a$, as $n \to \infty$.

Example of an Attractive Fixed Point

The square root function $g(x) = \sqrt{x}$ has a fixed point at $x = 1$.

For any initial value $x > 0$, if we iteratively apply $g(x)$ over and over again, the iterated function sequence

$$x, \sqrt{x}, \sqrt{\sqrt{x}}, \sqrt{\sqrt{\sqrt{x}}}, \ldots$$

converges to 1.

So $x = 1$ is an attractive fixed point of $g(x) = \sqrt{x}$.

Fixed Point Iteration

The **fixed point iteration** algorithm is described as follows:

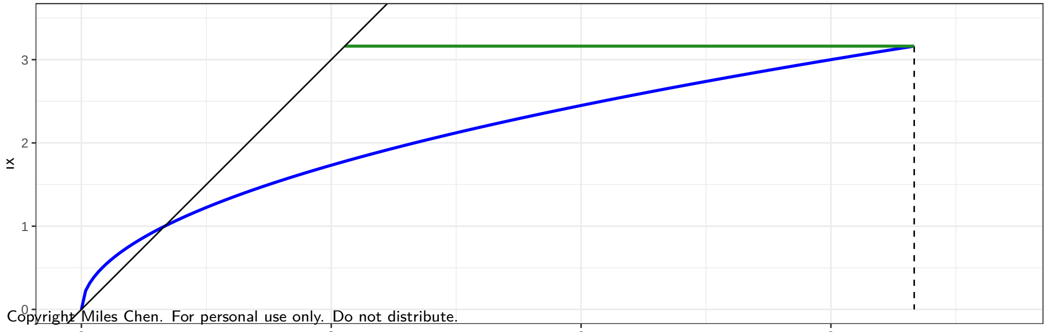
- Start from an initial value x_0.
- Compute $g(x_0)$ as the next value x_1.
- Repeat:

$$x_{n+1} = g(x_n)$$

To avoid iterating forever, we stop when $|x_n - x_{n-1}| \leq \varepsilon$, for some user-specified **tolerance** level $\varepsilon > 0$.

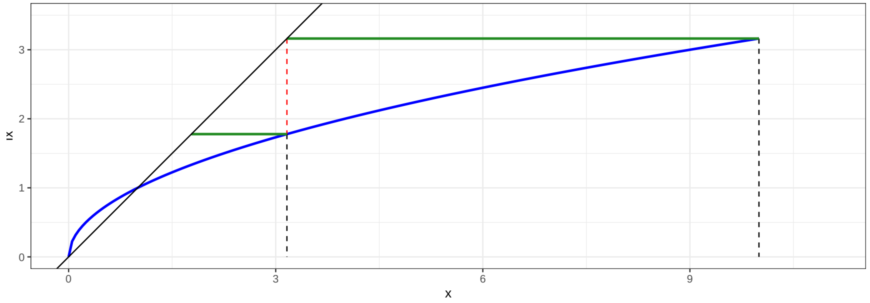
Fixed Point Iteration

```
Starting value is: 10
Next value of x is: 3.162278
```



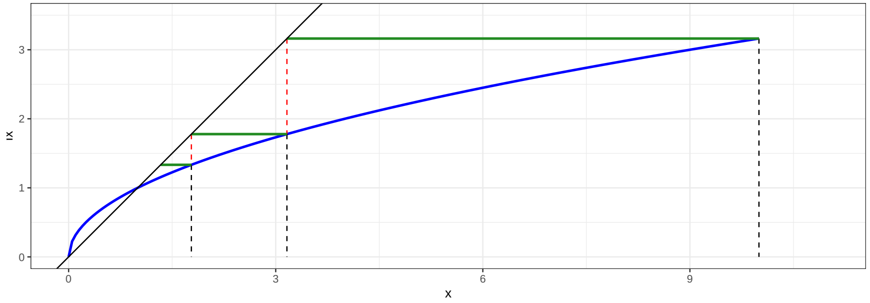
Fixed Point Iteration for sqrt(x)

```
Next value of x is: 1.778279
```



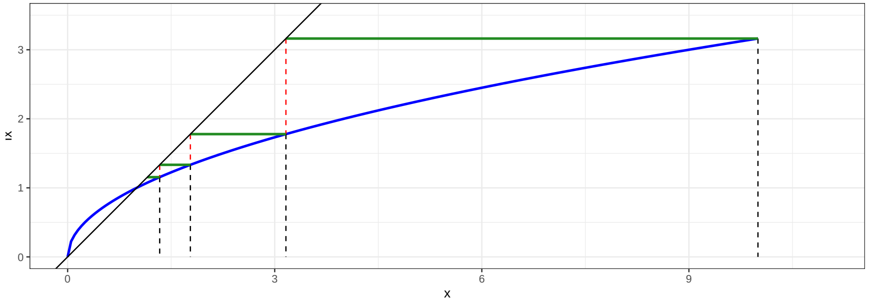
Fixed Point Iteration for sqrt(x)

```
Next value of x is: 1.333521
```



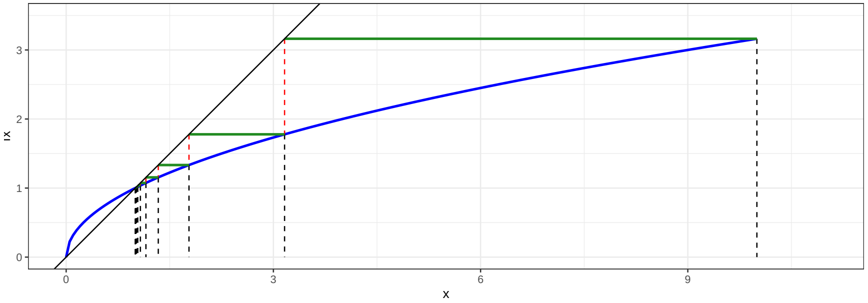
Fixed Point Iteration for sqrt(x)

```
Next value of x is: 1.154782
```



Fixed Point Iteration for sqrt(x)

After 15 iterations, $x \approx 1.00007$



Fixed Points for Root Finding

The problem of finding fixed points can be used to find roots of a function.

Let's say there is a function $f(x)$ for which you wish to find roots.

The goal is to define another function $g(x)$ so that when $g(x) = x$, then $f(x) = 0$.

For example, one possibility is to let $g(x) = x - f(x)$. Then $f(x) = 0$ if and only if $g(x) = x$

With this we can find roots of $f(x)$ by finding the fixed points of $g(x)$.

Note: There are often many ways to write $f(x) = 0$ as $g(x) = x$, and, in practice, some are better than others.

Example of Fixed Point Iteration for Root Finding

Suppose we want to find the solution to the equation

$$f(x) = \log(x) - e^{-x} = 0$$

We first want to express this as a fixed point problem.

One way to rewrite $f(x)$ is to solve for x:

$$\begin{aligned} \log(x) - e^{-x} &= 0 \\ \log(x) &= e^{-x} \\ x &= e^{e^{-x}}. \end{aligned}$$

Let $g(x) = e^{e^{-x}}$. When $g(x) = x$, then $f(x) = 0$

Example of Fixed Point Iteration

An implementation of fixed point iteration for $g(x) = e^{e^{-x}}$:

```r
g <- function(x) {
  exp(exp(-x))
}
```

```r
x_old <- 10
x_new <- g(x_old)
iterations <- 0
tol <- 1e-8
while (abs(x_new - x_old) > tol) {
  x_old <- x_new
  x_new <- g(x_old)
  iterations <- iterations + 1
}
```

Note: For algorithms that might not always converge (or are possibly slow to converge), it is good practice to have an option for stopping the algorithm after a specified maximum number of iterations. This prevents your code from running forever.

Example of Fixed Point Iteration

$$f(x) = \log(x) - e^{-x} = 0$$

Our results:

```
x_new # approximate root
```

```
## [1] 1.3098
```

```
log(x_new) - exp(-x_new) # f(x) at x_new is approx 0
```

```
## [1] 9.711432e-10
```

```
iterations
```

```
## [1] 19
```

Fixed Point Convergence

If we assume the fixed point iteration method converges, i.e., $x_{n+1} = g(x_n)$ and $x_n \to a$, we can show that the value it converges to a is a fixed point of $g(x)$.

Given g is continuous, then

$$a = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} g(x_n) = g\left(\lim_{n\to\infty} x_n\right) = g(a),$$

so a is a fixed point of g.

That is, *if* the fixed point iteration method converges, then it converges to a fixed point.

But does fixed point iteration always converge?

Fixed Point Convergence

The convergence of fixed point iteration relies on the existence of an attractive fixed point. If the fixed point is attractive, and the initial value is relatively "close" to the fixed point, then the fixed point iteration method is guaranteed to converge.

Warning: Not all functions have fixed points, and not all fixed points are attractive. Thus, in general, the fixed point iteration method is not guaranteed to converge.

A sufficient condition depends on the value of the *derivative* $g'(x)$ at the fixed point: If a is a fixed point of g and $|g'(a)| < 1$, then a is attractive.

If $|g'(a)| > 1$, then the fixed point is called **repelling**, and the fixed point iteration method will diverge.

Example of Fixed Point Iteration - Diverging

Let's revisit the same equation from before

$$f(x) = \log(x) - e^{-x} = 0$$

We could have rewriten $f(x) = 0$ as $g(x) = x$ in another way

$$\begin{aligned} \log(x) - e^{-x} &= 0 \\ \log(x) - e^{-x} + x &= x \end{aligned}$$

Let $g(x) = \log(x) - e^{-x} + x$. When $g(x) = x$, then $f(x) = 0$

Example of Fixed Point Iteration - Diverging

```r
g <- function(x) {
  log(x) - exp(-x) + x
}
```

```r
x_old <- 10
x_new <- g(x_old)
iterations <- 0
tol <- 1e-8
while (abs(x_new - x_old) > tol) {
  x_old <- x_new
  x_new <- g(x_old)
  iterations <- iterations + 1
  if (iterations > 1000) {break} # iteration limit
}
```

Results

```
x_new # result
```

```
## [1] 7830.023
```

```
log(x_new) - exp(-x_new) # f(x) at x_new is way off
```

```
## [1] 8.965721
```

```
iterations # we reached our iteration limit
```

```
## [1] 1001
```

Section 3

Newton-Raphson Method

Newton-Raphson Method

Fixed point iteration is considered relatively slow, as the approximation error is usually divided by a constant factor at each iteration.

A well-known method for much faster convergence that is still widely used today is the **Newton-Raphson** method.

The main idea behind Newton-Raphson is to use the property that the tangent line at a point is the best linear approximation to the function near that point.

Linear Approximation

Let a be a root of f. Suppose that f is differentiable with continuous derivative f'.

Let x_0 denote the initial guess at a. The **linear approximation** to $f(x)$ close to x_0 is the equation of the tangent line

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

Side Note (not on exams): This works because the Taylor series expansion of $f(x)$ around $x = x_0$, given by

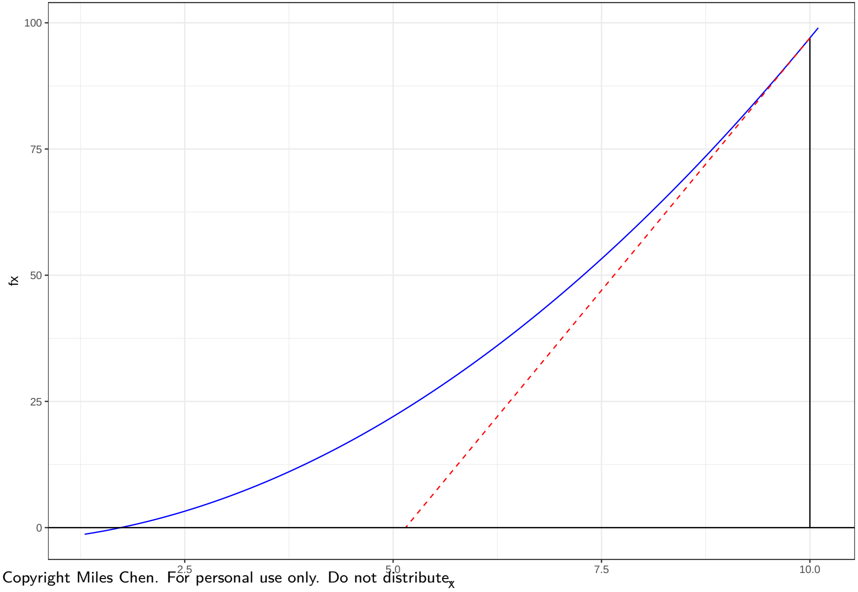
$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \cdots$$

We simply drop the higher order terms.

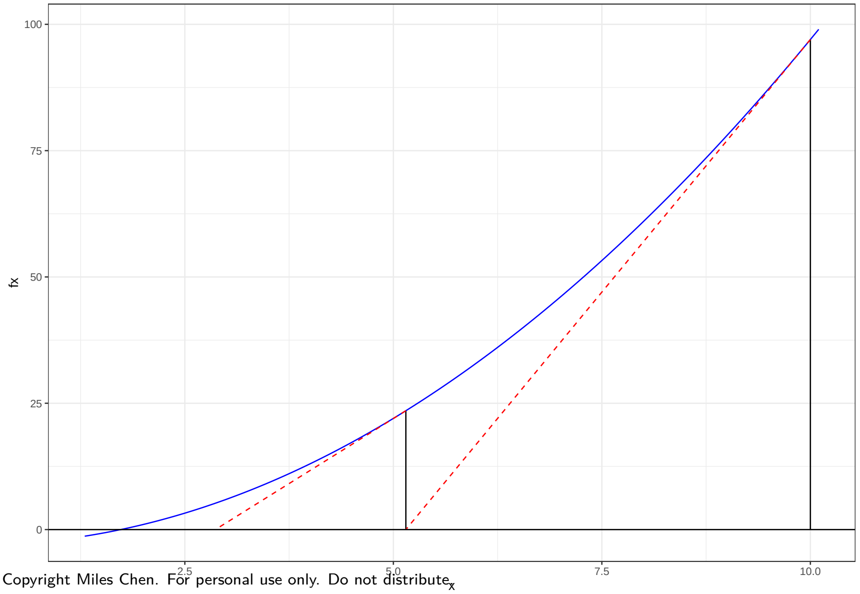
Linear Approximation

The tangent line is a good approximation to f near x_0. The next guess x_1 is the point where the tangent line crosses the x-axis.

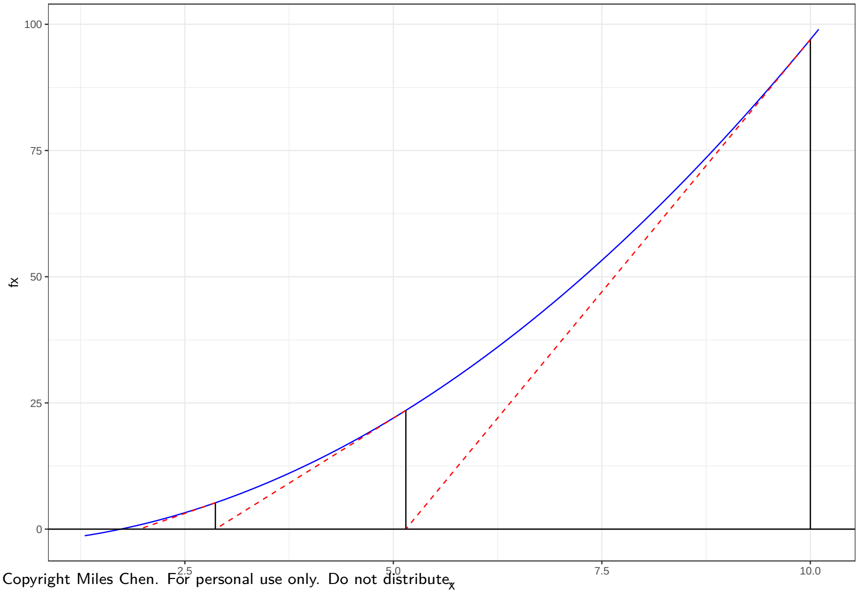
$f(x) = x^2 - 3$ Start point $x_0 = 10$



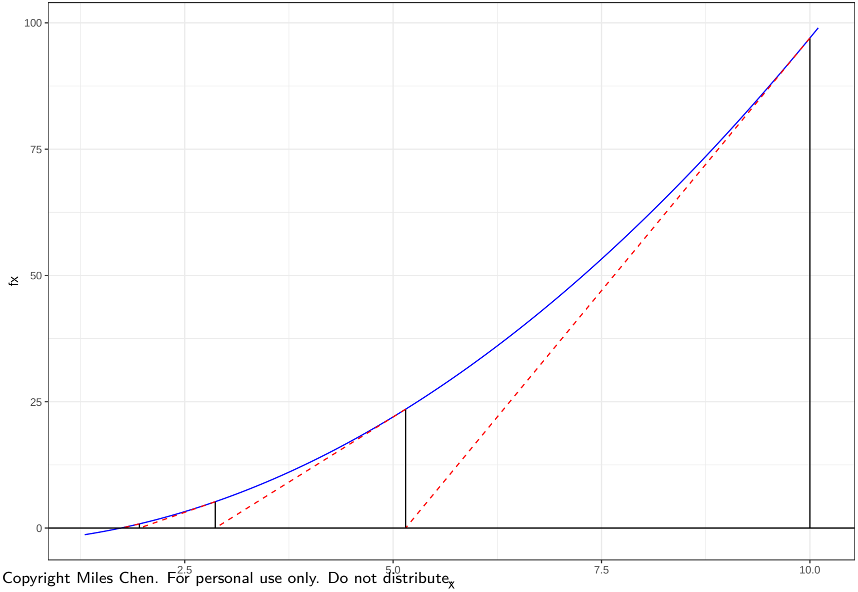
$f(x) = x^2 - 3,\ x_1 = 5.15$



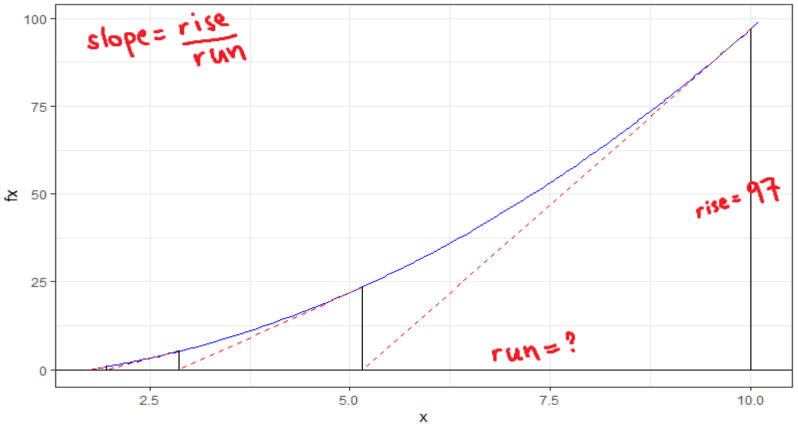
$f(x) = x^2 - 3$, $x_2 = 2.866262$



$f(x) = x^2 - 3$, 1.956461



$f(x) = x^2 - 3, x_0 = 10$



$slope = \frac{rise}{run}$. This can be rearranged to: $run = \frac{rise}{slope}$.

The rise is $f(x_0) = 97$. The slope is $f'(x_0) = 20$. The $run = \frac{97}{20} = 4.85$

Linear Approximation

$f(x) = x^2 - 3$

Start point is $x_0 = 10$, rise $= f(10) = 97$, and slope $= f'(10) = 20$

The tangent line will cross the x-axis at $10 -$ run $= 10 - \frac{97}{20} = 5.15$

That is to say:

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

Newton-Raphson Method

Repeating the same procedure, the next guess x_2 is then

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)},$$

and, in general,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Newton-Raphson Method

The **Newton-Raphson** algorithm is described as follows:

- Start from an initial value x_0.
- Repeat:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Stop when $|f(x_n)| \leq \varepsilon$, where $\varepsilon > 0$ is the pre-specified tolerance level.

Example of Newton-Raphson Method

Suppose we want to use the Newton-Raphson method to find the root of

$$f(x) = \log(x) - e^{-x}.$$

Taking the derivative with respect to x, we have

$$f'(x) = \frac{1}{x} + e^{-x}.$$

Example of Newton-Raphson Method

An implementation of Newton-Raphson for $f(x) = \log(x) - e^{-x}$:

```r
f <- function(x) {
  log(x) - exp(-x)
}
f_prime <- function(x) {
  (1 / x) + exp(-x)
}
```

```r
x <- 2
iterations <- 0
tol <- 1e-8
while (abs(f(x)) > tol) {
  x <- x - f(x) / f_prime(x)
  iterations <- iterations + 1
}
```

Example of Newton-Raphson Method

```r
x # approximate root
```

```
## [1] 1.3098
```

```r
log(x) - exp(-x) # indeed, f(x) is close to 0
```

```
## [1] -3.494008e-09
```

```r
iterations # only took 4 iterations to reach this value
```

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

Newton-Raphson Convergence

If we assume that the Newton-Raphson method converges, i.e., $x_n \to a$, we can show that the value it converges to a is a root. Given f is differentiable and f' is continuous, then

$$\begin{aligned} a &= \lim_{n\to\infty} x_{n+1} &= \lim_{n\to\infty}\left[x_n - \frac{f(x_n)}{f'(x_n)}\right] \\ && = \lim_{n\to\infty} x_n - \frac{\lim_{n\to\infty} f(x_n)}{\lim_{n\to\infty} f'(x_n)} \\ && a = a - \frac{f(a)}{f'(a)}. \end{aligned}$$

Thus, assuming $f'(a) \neq \pm\infty$, then $f(a) = 0$ (i.e., a is a root of f).

That is, *if* the Newton-Raphson method converges, and the derivative at a is finite, then it converges to a root.

But does Newton-Raphson always converge?

Newton-Raphson Convergence

It can be shown that if f is “well behaved” at a (i.e., $f'(a) \neq 0$ and f'' is finite and continuous at a), and you start with an initial value x_0 that is “close enough” to a, then x_n will converge quickly.

In fact, in most cases, the Newton-Raphson method converges **quadratically**, meaning the number of correct digits doubles per iteration.

However, when the conditions for quadratic convergence are not satisfied, then the Newton-Raphson method is not guaranteed to converge as quickly, and it may not even converge at all.

Newton-Raphson Convergence

The Newton-Raphson method can fail for several reasons.

- If $f'(x_n) = 0$ for any n, then the tangent line is horizontal and will not cross the x-axis. There can also be issues even if $f'(x_n)$ is not 0 but close to 0. (The derivative $f'(x)$ is allowed to be zero at certain locations, but it cannot be zero at any of the iterative locations x_n.)
- Convergence can depend on the initial guess. If the starting value is too far from the root, the method may either not converge or converge to a different root.
- Sometimes iterative locations can oscillate back and forth in a cycle, and the Newton-Raphson method will not converge.
- If the second derivative $f''(x)$ is infinite or undefined at the root, it can also cause non-convergence.
- In some non-ideal scenarios (e.g., when the derivative at the root is zero, $f'(a) = 0$), the Newton-Raphson method may still converge, but convergence may be much slower.

Animated examples: https://yihui.name/animation/example/newton-method/

Section 4

Secant Method

Secant Method

Because of its fast convergence, the Newton-Raphson method is often the preferred method for root-finding when applicable.

One immediate downside to Newton-Raphson, however, is that it requires the analytic computation of the derivative of $f(x)$.

If the derivative is difficult to compute or does not exist, then we need a different method. An often used substitute method is the **secant method**.

Like the Newton-Raphson method, the secant method is based on a linear approximation to the function of interest.

Secant method

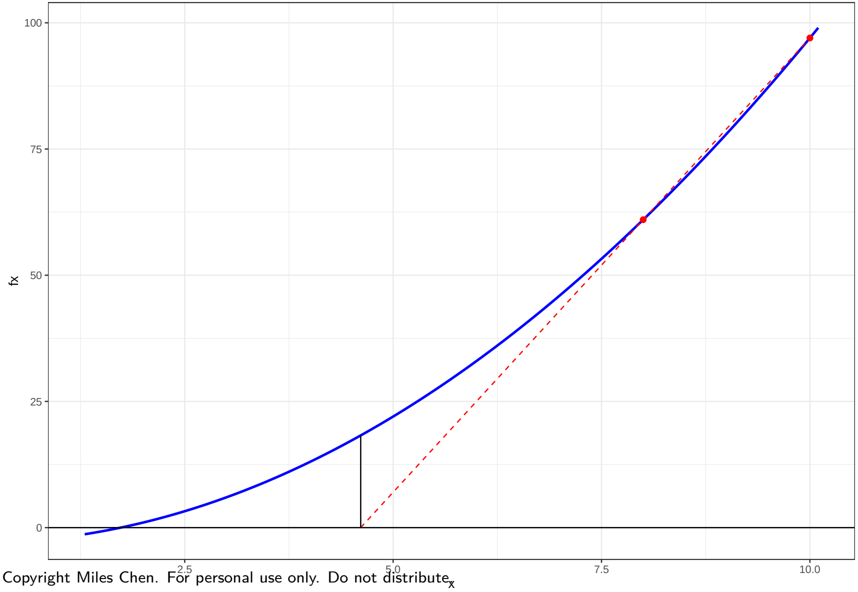
Let a be a root of f. We start with *two* initial guesses, x_0 and x_1, for the value of a. It does not matter if one guess is larger or smaller than a or if both guesses are both larger or both smaller than a.

We draw a straight line that connects the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$.

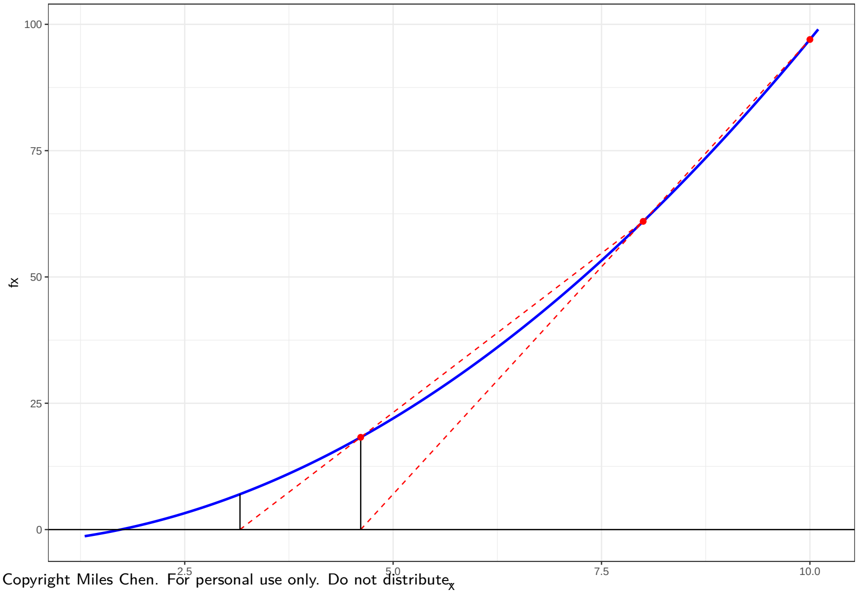
This straight line is called a **secant line** to the curve $y = f(x)$. In a neighborhood around x_0 and x_1, the secant line is a linear approximation to $f(x)$.

We see where the secant line crosses the x-axis, and that location will be the next value we use.

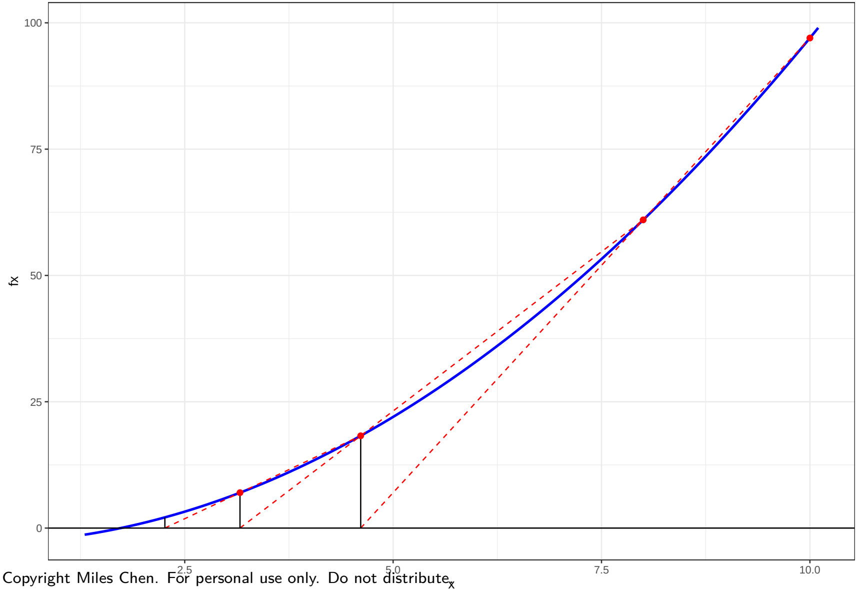
$f(x) = x^2 - 3, x_0 = 10, x_1 = 8$



$f(x) = x^2 - 3, x_0 = 10, x_1 = 8$



$f(x) = x^2 - 3, x_0 = 10, x_1 = 8$



Finding the Next value to use

Our first guess is $x_0 = 10$ with $f(10) = 10^2 - 3 = 97$

The second guess is $x_1 = 8$ with $f(8) = 8^2 - 3 = 61$

The slope that connects these two lines is

$$\frac{\text{change is y}}{\text{change in x}} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{(61 - 97)}{(8 - 10)} = 18$$

The line passes through the point $(8, 61)$.

In point slope form, the equation of this line is:

$$y - f(x_1) = m(x - x_1) \implies y - 61 = 18(x - 8)$$

Solve for the next value of x

In point slope form, the equation of this line is:

$$y - f(x_1) = m(x - x_1) \implies y - 61 = 18(x - 8)$$

We solve this equation for where $y = 0$ and we get:

$$0 - 61 = 18(x - 8)$$
$$8 - 61 \cdot \frac{1}{18} = x$$

So the next value we will use as a ‘guess’ is 4.6111.

Linear Approximation with the Secant Line

More generally:

The slope of the secant line is

$$m = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

so the equation of the secant line, in point-slope form, is

$$y - f(x_1) = m(x - x_1)$$

$$y - f(x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1)$$

Linear Approximation with the Secant Line

Since the secant line is a good approximation to f near x_1, the next guess x_2 is then the point where the secant line crosses the x-axis.

That is, we choose x_2 so that $y = 0$.

$$\begin{aligned}
y - f(x_1) &= \tfrac{f(x_1)-f(x_0)}{x_1-x_0}(x - x_1) \\
0 - f(x_1) &= \tfrac{f(x_1)-f(x_0)}{x_1-x_0}(x_2 - x_1) \\
-f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)} &= x_2 - x_1 \\
x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)} &= x_2
\end{aligned}$$

Secant Method

In general:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Note: Notice that if x_n and x_{n-1} are close to each other, then

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}},$$

which is why the secant method is often seen as an approximation to the Newton-Raphson method, where we approximate the derivative by a finite difference approximation.

Secant Method

The **secant method** algorithm is described as follows:

- Start from initial values x_0 and x_1.
- Repeat:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Stop when $|f(x_n)| \le \varepsilon$, where $\varepsilon > 0$ is the pre-specified tolerance level.

Secant Method Convergence

The convergence properties of the secant method are similar to those of the Newton-Raphson method.

If f is well behaved at a, and you start with initial values x_0 and x_1 sufficiently close to a, then x_n will converge to a quickly, though not quite as fast as the Newton-Raphson method.

Just as for the Newton-Raphson method, we cannot guarantee convergence for the secant method.

The main trade-offs in using the secant method over Newton-Raphson: We no longer need to know f', but we lose some speed and have to provide two initial points, x_0 and x_1.

Section 5

Bisection Method

Bisection Method

The **bisection method** is one of the simplest numerical methods for root-finding to implement, and it almost always works. The main disadvantage is that convergence is relatively slow.

The idea behind the bisection method is **root-bracketing**, which works by first finding an interval in which the root must lie, and then successively refining the bounding interval in such a way that the root is guaranteed to always lie inside the interval.

In the bisection method, the width of the bounding interval is successively halved.

Bisection Method

Suppose that f is a continuous function. Then f has a root in the interval (x_ℓ, x_r) if either:

- $f(x_\ell) < 0$ and $f(x_r) > 0$, or
- $f(x_\ell) > 0$ and $f(x_r) < 0$.

A convenient way to verify this condition is to check if

$$f(x_\ell)f(x_r) < 0$$

That is to say, if $f(x_\ell)$ and $f(x_r)$ have opposite signs, then a root must exist between x_ℓ and x_r

The bisection method works by taking an interval (x_ℓ, x_r) that contains a root, then successively refining x_ℓ and x_r until $x_r - x_\ell \leq \varepsilon$, where $\varepsilon > 0$ is the pre-specified tolerance level.

Bisection Method

The **bisection** algorithm is described as follows:

Start with $x_\ell < x_r$ such that $f(x_\ell)f(x_r) < 0$.

1. If $x_r - x_\ell \leq \varepsilon$, then stop.
2. Compute $x_m = \dfrac{x_\ell + x_r}{2}$. If $f(x_m) = 0$, then stop.
3. If $f(x_\ell)f(x_m) < 0$, then assign $x_r = x_m$.
   Otherwise, assign $x_\ell = x_m$.
4. Go back to step 1.

Notice that, at every iteration, the interval (x_ℓ, x_r) always contains a root. Assuming we start with $f(x_\ell)f(x_r) < 0$, the algorithm is guaranteed to converge, with the approximation error reducing by a constant factor $1/2$ at each iteration. If we stop when $x_r - x_\ell \leq \varepsilon$, then we know that both x_ℓ and x_r are within ε distance of a root.

Example of Bisection Method

An implementation of bisection method for $f(x) = \log(x) - e^{-x}$:

```r
f <- function(x) {
  log(x) - exp(-x)
}
```

```r
x_l <- 1
x_r <- 2
tol <- 1e-8
f_l <- f(x_l)
f_r <- f(x_r)
f_l * f_r < 0
```

```
## [1] TRUE
```

Example of Bisection Method

```r
its <- 0
while (x_r - x_l > tol) {
  x_m <- (x_l + x_r) / 2
  f_m <- f(x_m)
  if (identical(all.equal(f_m, 0), TRUE)) {
    break
  }
  if (f_l * f_m < 0) {
    x_r <- x_m
  } else {
    x_l <- x_m
  }
  its <- its + 1
}
(x_l + x_r) / 2
```

```
## [1] 1.3098
```

```r
its
```

Best of Both Worlds

The most popular current root-finding methods use root-bracketing to get close to a root, then switch over to the Newton-Raphson or secant method when it seems safe to do so.

This strategy combines the safety of bisection with the speed of the secant method.

Order of Convergence

Let a be the root and x_n be the nth approximation to the root. Define the **error** to be

$$\varepsilon_n = a - x_n.$$

A root finding method has an **order of convergence** p if, for large n, we have the approximate relationship

$$|\varepsilon_{n+1}| = k|\varepsilon_n|^p,$$

for some positive constant k. Larger values of p correspond to faster convergence.

Side Note: The derivations for the orders of convergence discussed here are in Jeffrey R. Chasnov's lecture notes on numerical methods: https://www.math.ust.hk/~machas/numerical-methods.pdf

Order of Convergence

The order of convergence of the bisection method is 1, since the error is reduced by approximately a factor of 2 with each iteration. That is,

$$|\varepsilon_{n+1}| = \frac{1}{2}|\varepsilon_n|.$$

Order of convergence 1 is also called **linear convergence**.

It can be shown that, under certain conditions, the Newton-Raphson method has order of convergence 2, i.e.,

$$|\varepsilon_{n+1}| = k|\varepsilon_n|^2,$$

which is also called **quadratic convergence**.

The secant method can be shown to have an order of convergence equal to

$$p = \frac{1+\sqrt{5}}{2} \approx 1.618034,$$

which is the famous **golden ratio** (sometimes denoted by Φ)!