Newton's Method for Root Finding and Optimization: One Dimension

STAT 689: Statistical Computing

February 15, 2018

Newton's Method for Root Finding

Newton's Method for Optimization

Example: Extinction Probabilities

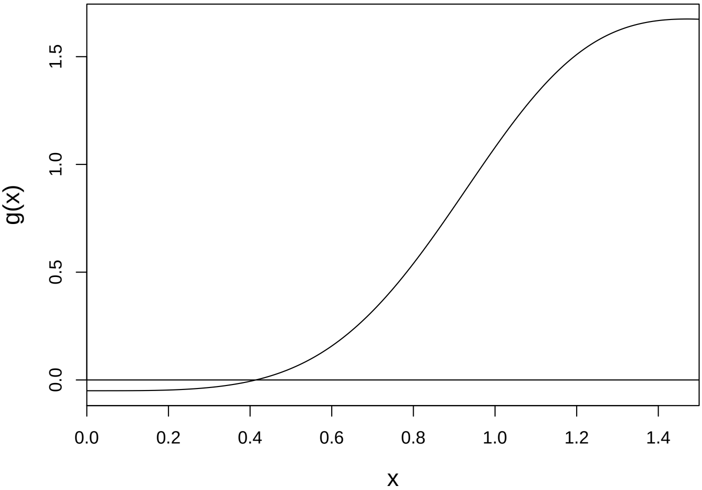
Outline

Newton's Method for Root Finding

Newton's Method for Optimization

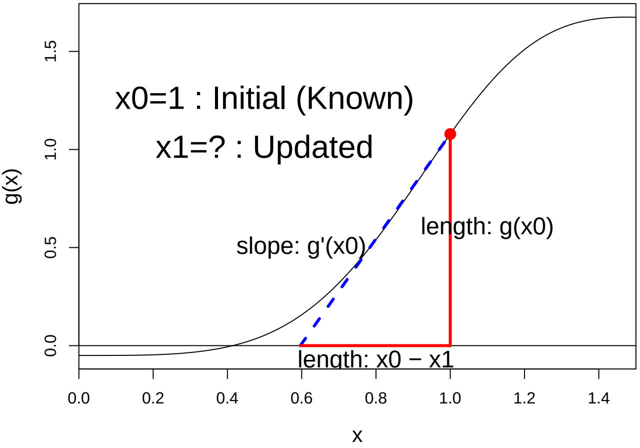
Example: Extinction Probabilities

Goal: Find Root of g



$$g(x) = 1.95 - e^{-2/x} - 2e^{-x^4}$$

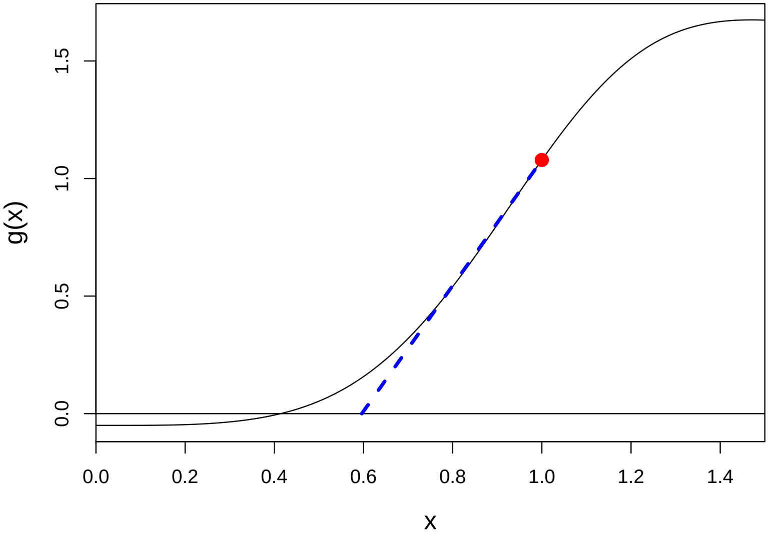
Newton: Approximate as Linear Near Root



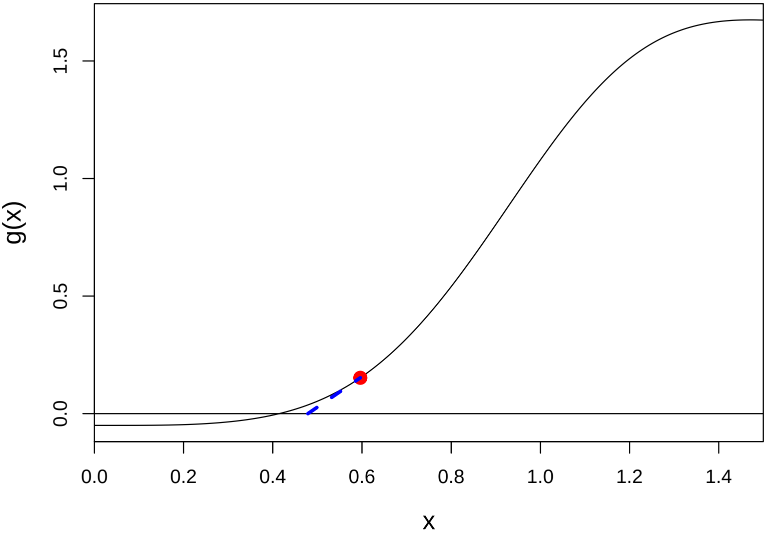
So

$$g'(x_0)(x_0 - x_1) = g(x_0)$$
$$x_1 = x_0 - g(x_0)/g'(x_0)$$

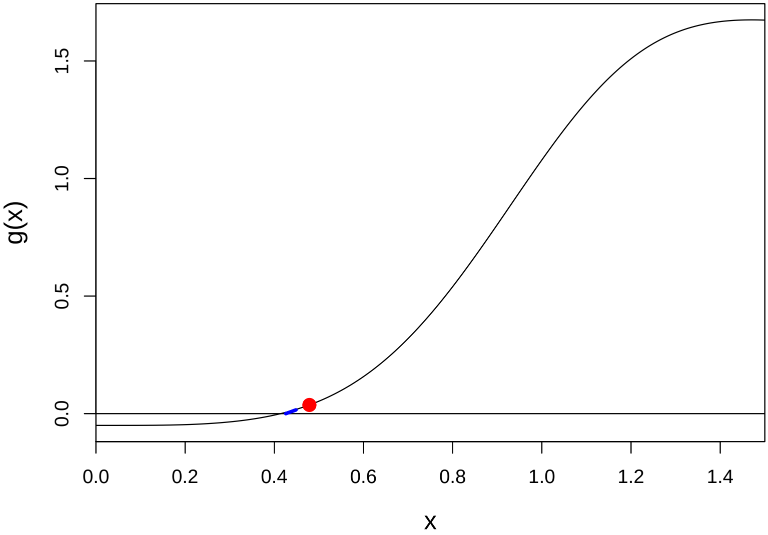
Iteration 1: Initial guess $x_0 = 1$



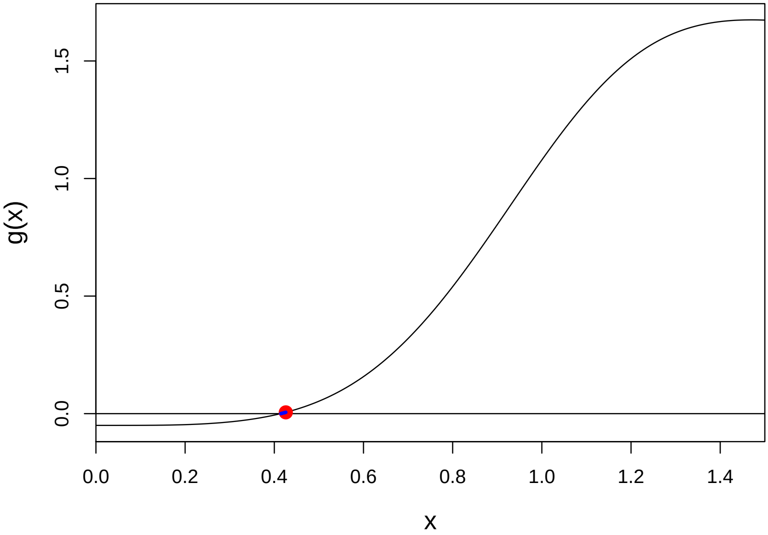
Iteration 2



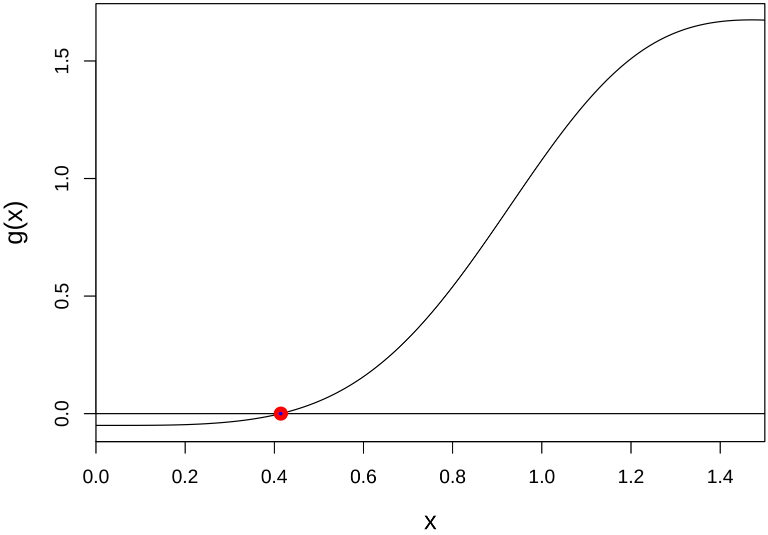
Iteration 3



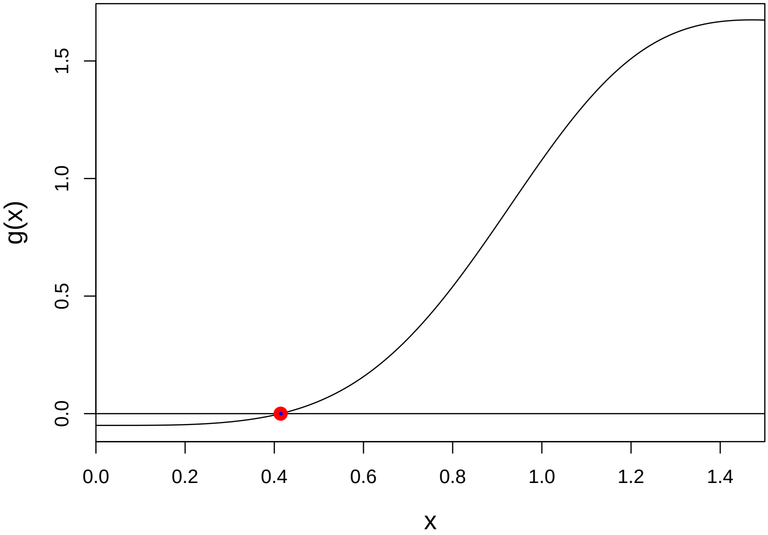
Iteration 4



Iteration 5



Iteration 6



Formulation

General formulation

$$x_n = x_{n-1} - g(x_{n-1})/g'(x_{n-1})$$

Here

$$g(x) = 1.95 - e^{-2/x} - 2e^{-x^4}$$
$$g'(x) = -2x^{-2}e^{-2/x} + 8x^3e^{-x^4}$$

So iterations are very fast.

Speed of Convergence

Bisection:

- start with interval $[a, b]$
- interval is cut in half at each iteration
- width of interval at iteration n is $2^{-n}(b-a)$
- let e_n be error at iteration n:

$$e_n \approx e_{n-1}/2$$

- this is linear convergence

Newton:

- Newton has quadratic convergence (see textbook)

$$e_n \approx Ce_{n-1}^2$$

- twice as many significant digits at each iteration

Coding Note in R

- R is a strongly functional language
- functions can go almost anywhere, including as arguments to functions
- very convenient for writing Newton's algorithm

```
## g = function to find root
g <- function(x) 1.95 - exp(-2/x) - 2*exp(-x^4)
## gd = derivative of g
gd <- function(x) -2*x^{-2}*exp(-2/x) + 8*x^3*exp(-x^4)
## update x
newton_update <- function(x,g,gd) x - g(x)/gd(x)
```

Outline

Newton's Method for Root Finding

Newton's Method for Optimization

Example: Extinction Probabilities

Use Newton's Method for Optimization

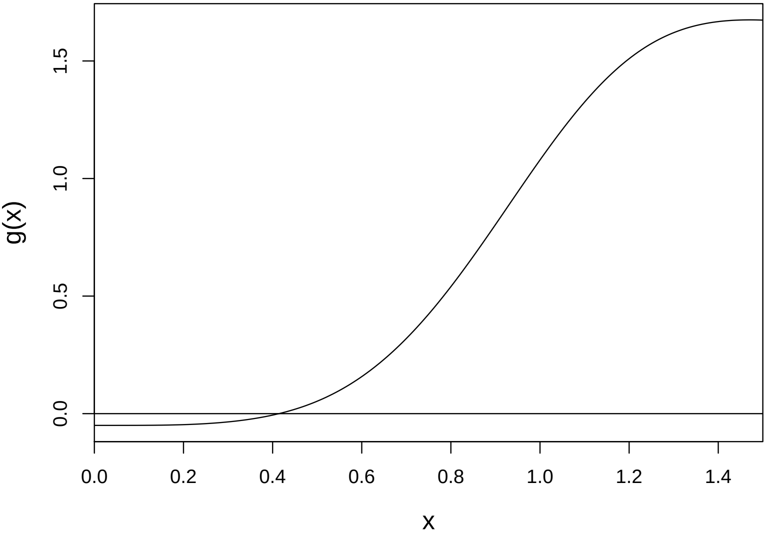
- in optimization, we find the (arg) maximum of g

$$x^* = \underset{x}{\operatorname{argmax}}\ g(x)$$

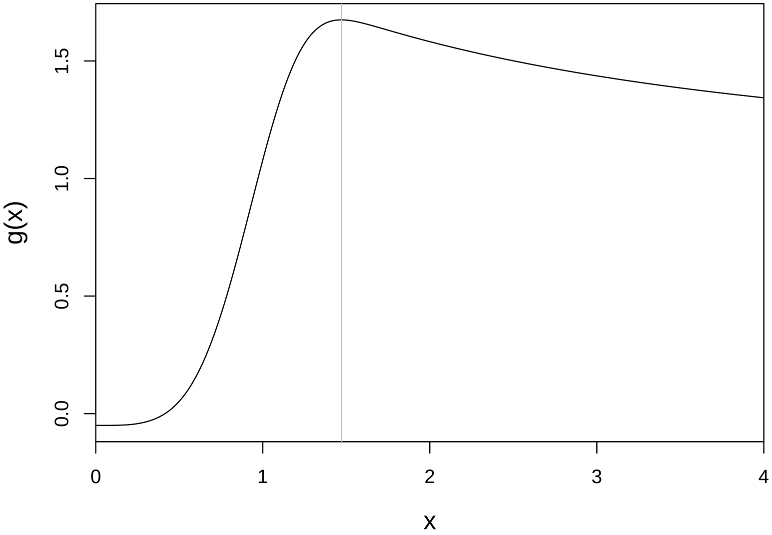
- this is often a root of g' (assuming differentiability, no domain constraints)
- apply newton's method to g'

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

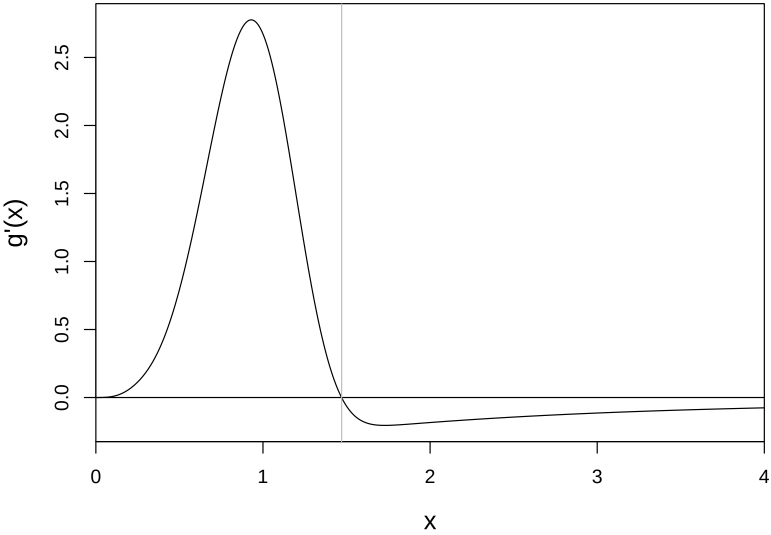
Consider the Same Function g



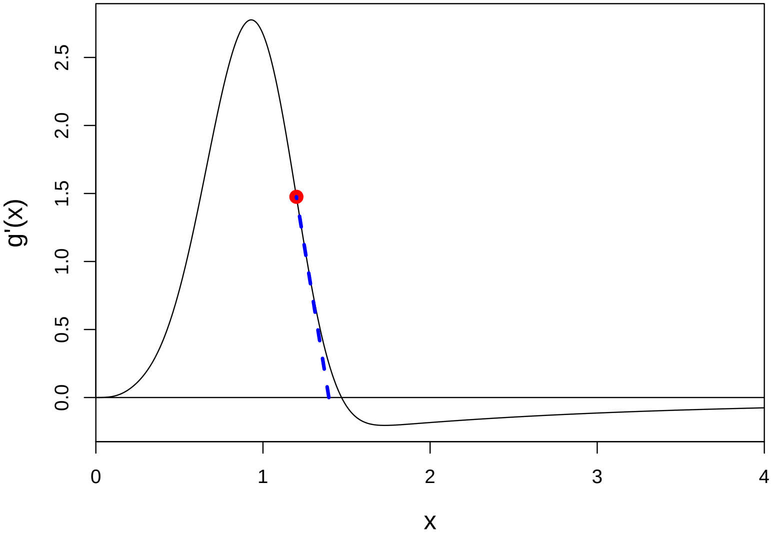
Zoom Out: Let's Find the Maximum with Newton



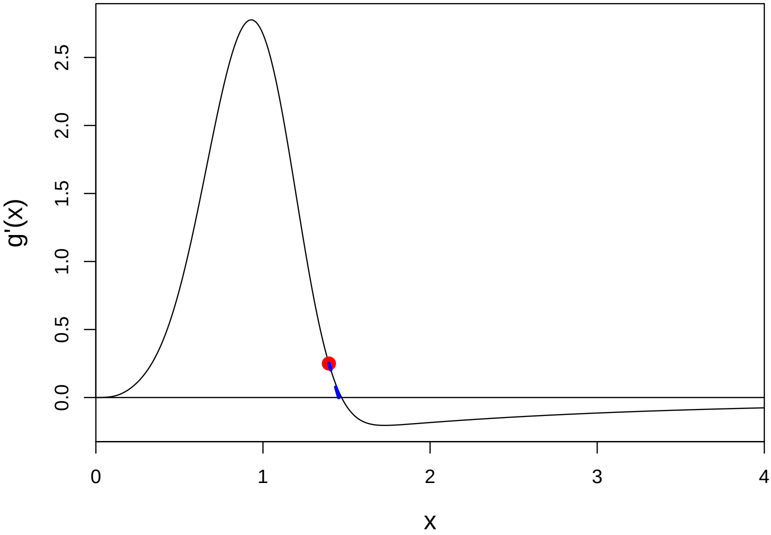
Compute g'



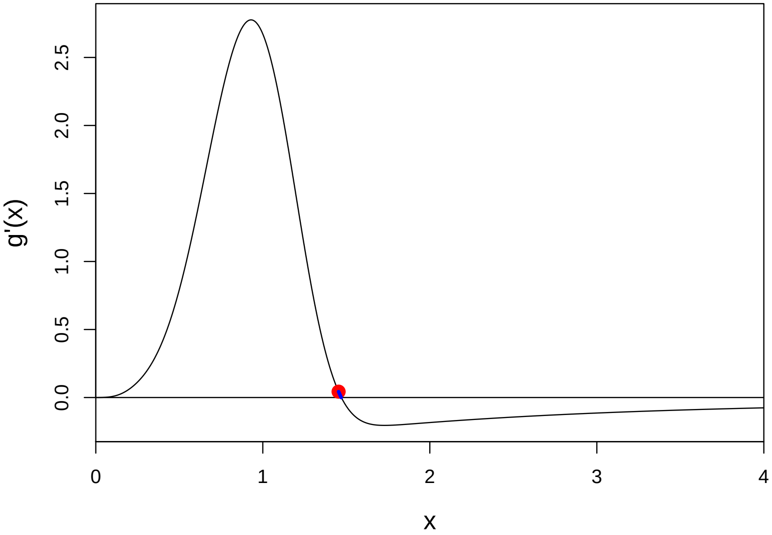
Iteration 1: Starting at $x_0 = 1.2$



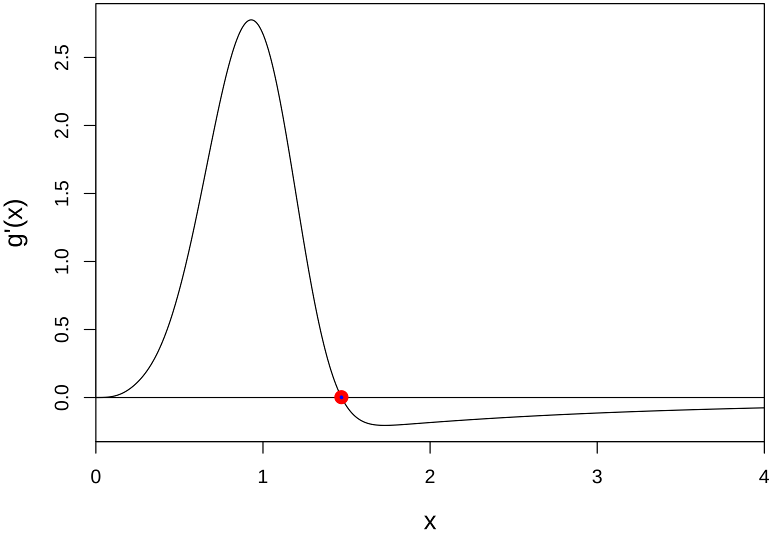
Iteration 2



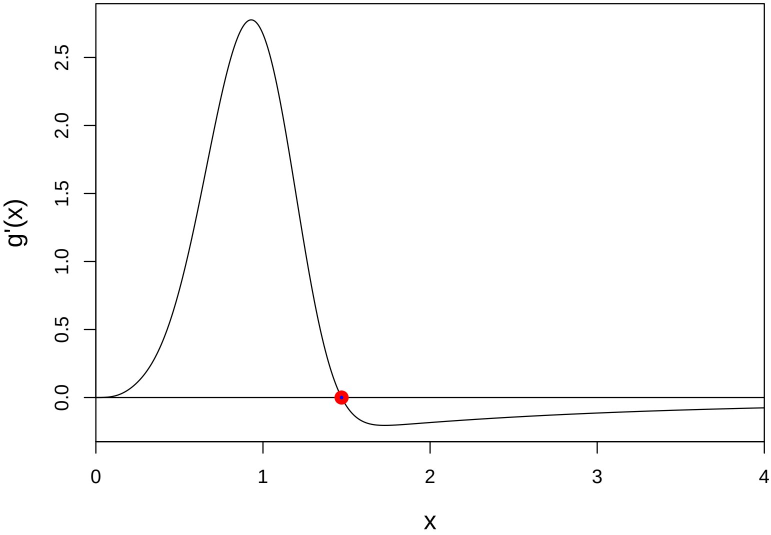
Iteration 3



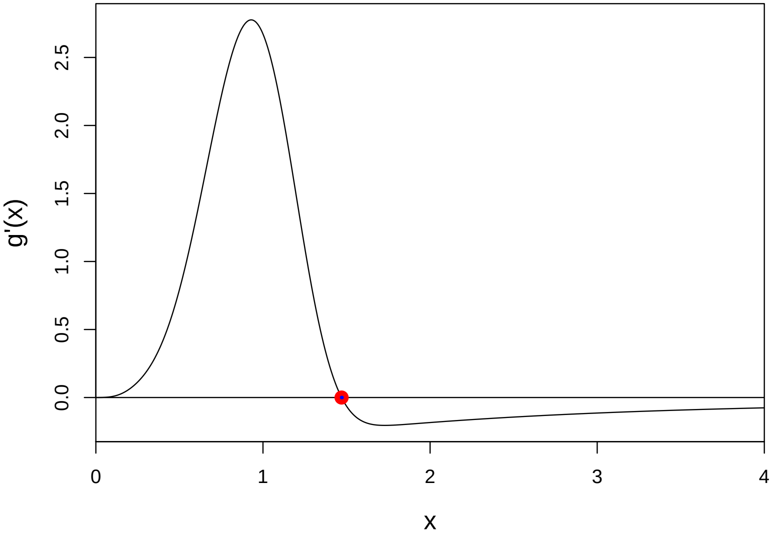
Iteration 4



Iteration 5



Iteration 6



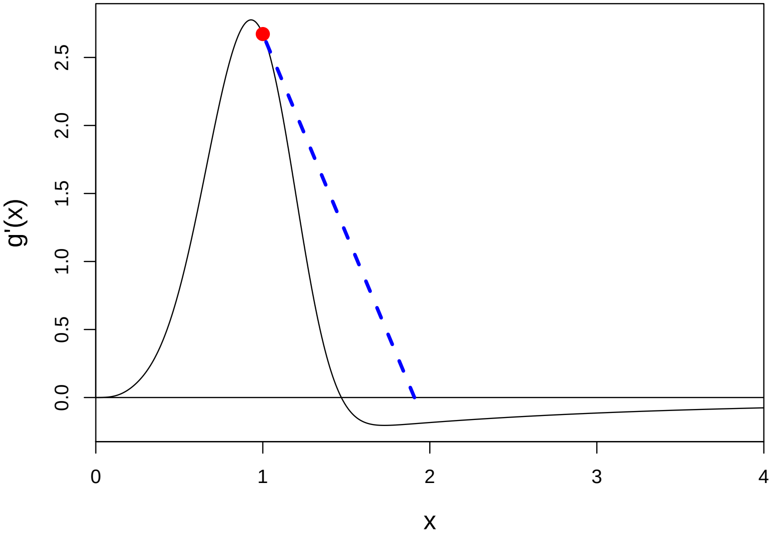
Convergence Problems

- Newton's method is looking for roots
  - could be local max
  - could be local / global min
- Newton can overshoot target (see next slide)

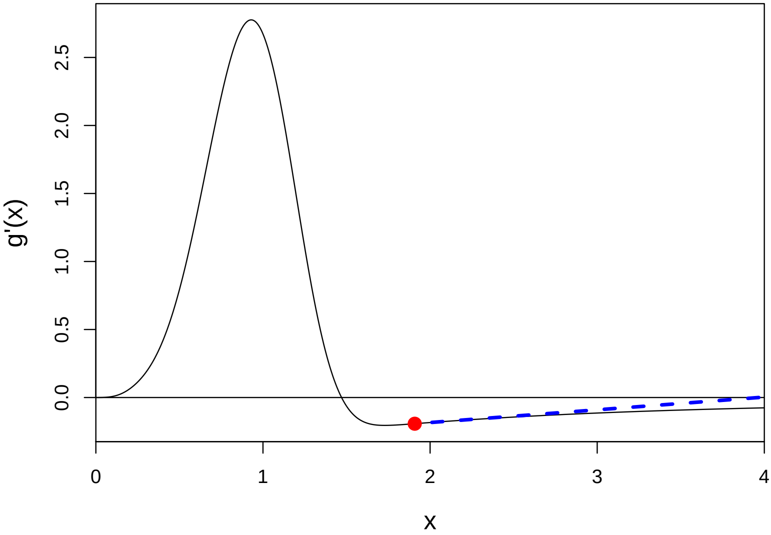
In general, need to start near solution and test that updates are increasing g.

We discuss these issues more in future lectures.

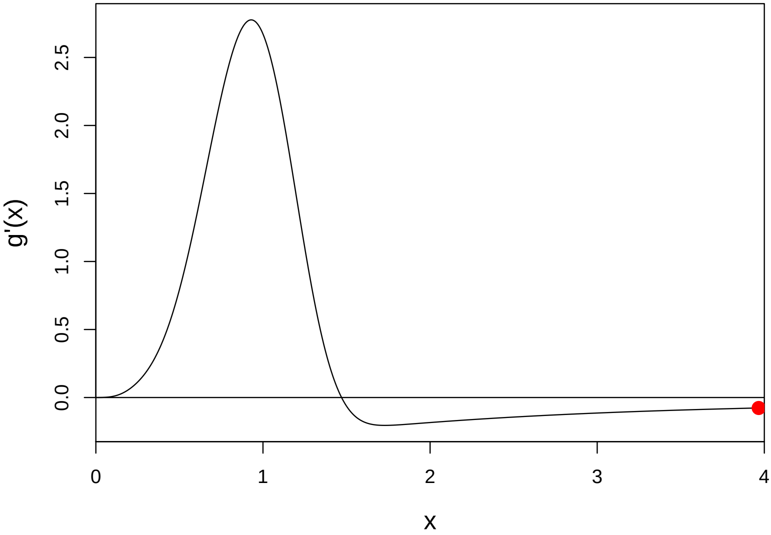
Iteration 1: Bad start at $x_0 = 1.0$



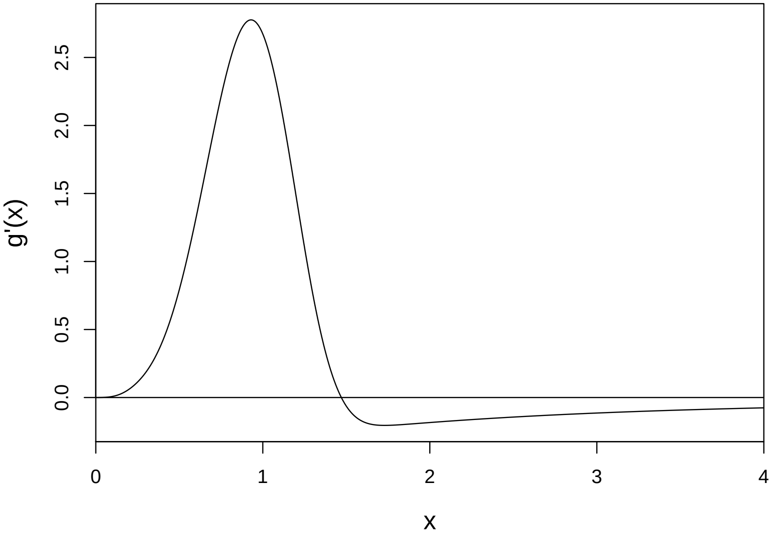
Iteration 2



Iteration 3



Iteration 4: The Red Dot Was Never Seen Again



Outline

Recall

- organism reproduces asexually
- probability of k offspring is p_k
- what is the probability that line will go extinct?

Define

- s_e be the probability of going extinct.
- n is maximum possible number of children, so $p_0, \ldots, p_n$.

Then

$$
\begin{aligned}
s_e &= \sum_{i=0}^{n} P(\text{extinct}|\text{i children}) P(\text{i children}) \\
&= \sum_{i=0}^{n} s_e^i p_i
\end{aligned}
$$

$P(\text{extinct}|\text{i children}) = s_e^i$ because if the first organism has i children, then the probability of going extinct is the probability that all the children's lines go extinct, s_e^i.

Equivalent Problem

Finding s_e is equivalent to finding root of

$$g(s) = \sum_{i=0}^{n} s^i p_i - s = p_0 + (p_1 - 1)s + \sum_{i=2}^{n} p_i s^i$$

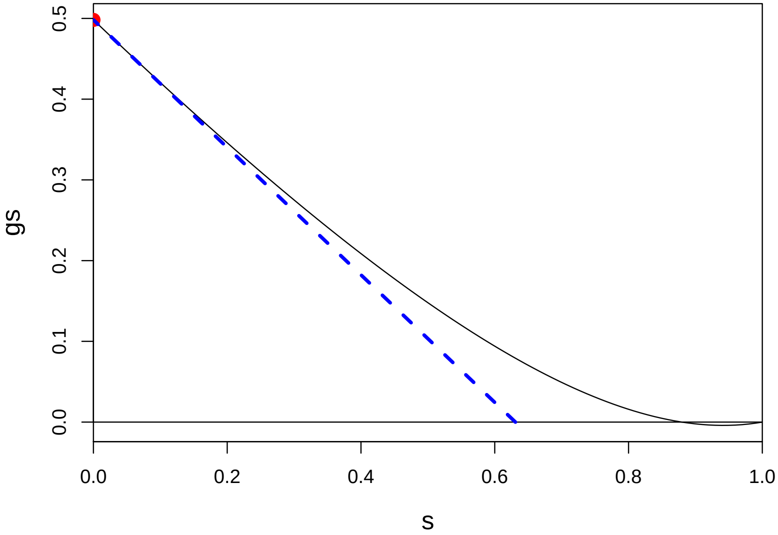
Easy to compute

$$g'(s) = \sum_{i=1}^{n} i s^{i-1} p_i - 1$$

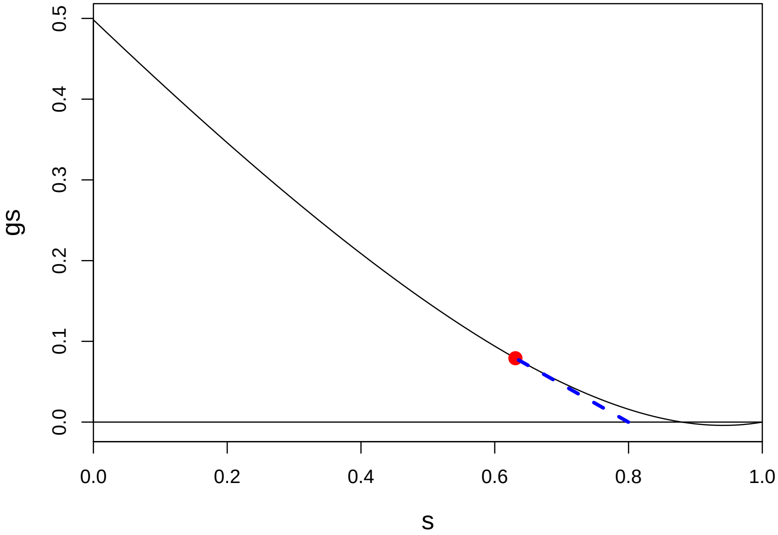
Consider the p coefficients from Lange:

```
p = [0.4982,0.2103,0.1270,0.0730,0.0418,
     0.0241,0.0132,0.0069,0.0035,0.0015,0.0005]
```

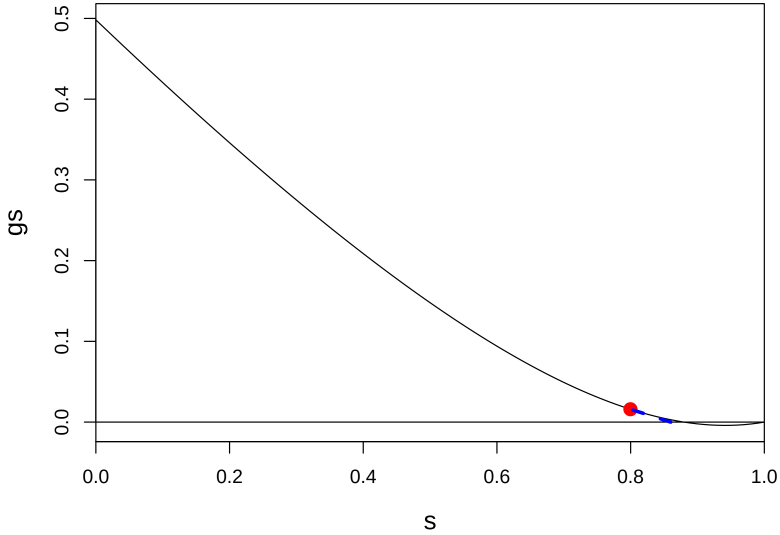
Iteration 1: Initial guess $x_0 = 0$



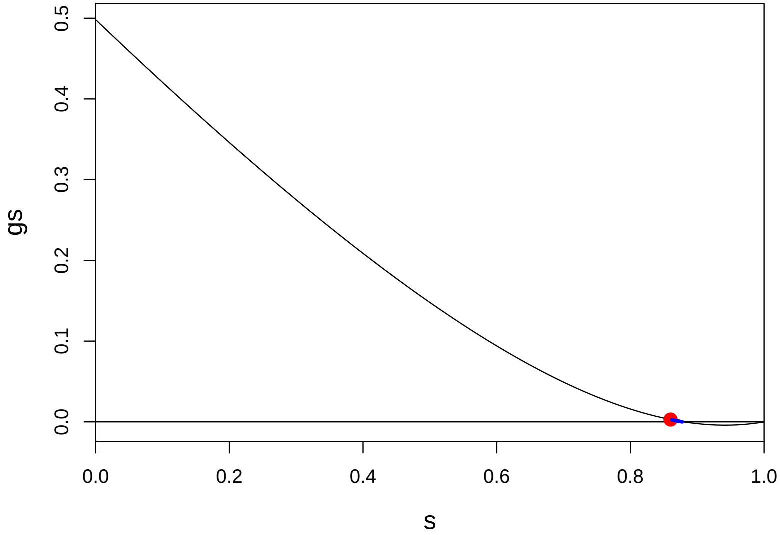
Iteration 2



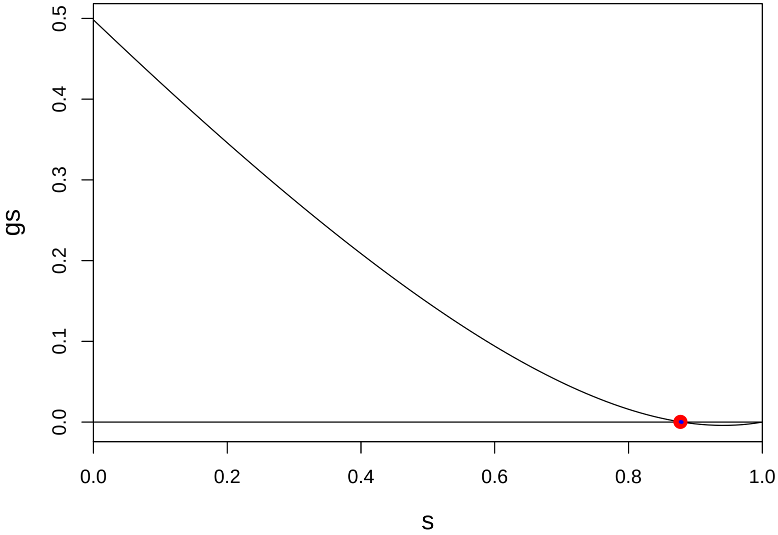
Iteration 3



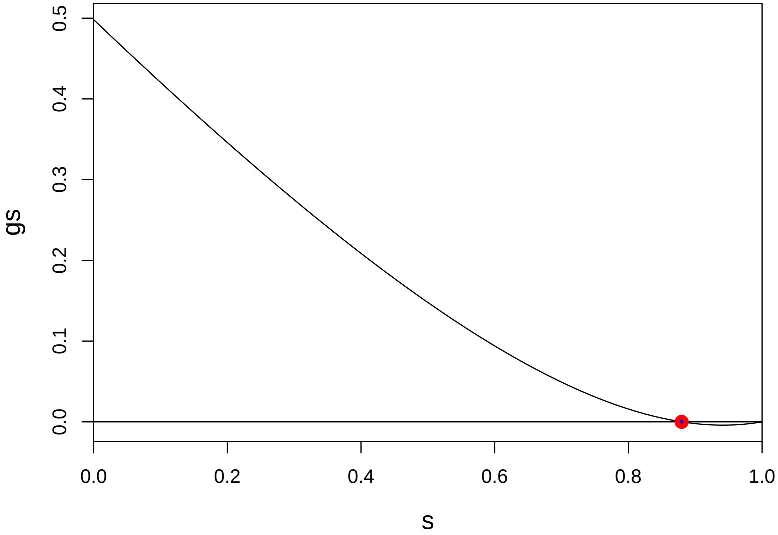
Iteration 4



Iteration 5



Iteration 6: $x_5 = 0.8797552$



R Code

```
rm(list=ls())
p = c(0.4982,0.2103,0.1270,0.0730,0.0418,
      0.0241,0.0132,0.0069,0.0035,0.0015,0.0005)
coeffs <- p
coeffs[2] <- coeffs[2] - 1

g <- function(s){
    return(sum(s^(0:(length(coeffs)-1))*coeffs))
}
gd <- function(s){
    n <- length(coeffs)
    return(sum((1:(n-1))*coeffs[2:n]*s^(0:(n-2))))
}

## perform newton updates
newton_update <- function(x,g,gd) x - g(x)/gd(x)
```

R Code

```
xn <- 0 ## initial guess
for(ii in 1:6){
    print(paste0("iteration: ",ii,
                 "  Estimate: ",round(xn,4)))
    xn <- newton_update(xn,g,gd)
}
```

```
## [1] "iteration: 1  Estimate: 0"
## [1] "iteration: 2  Estimate: 0.6309"
## [1] "iteration: 3  Estimate: 0.7997"
## [1] "iteration: 4  Estimate: 0.86"
## [1] "iteration: 5  Estimate: 0.8776"
## [1] "iteration: 6  Estimate: 0.8797"
```