Solution of non-linear equations

Finding MLEs, posterior modes (MAP estimates), minimizing loss functions, etc.

In many cases, this problem reduces to solving a nonlinear equation as

> arg min f(x) or arg max f(x) usually satisfies f'(x) = 0.

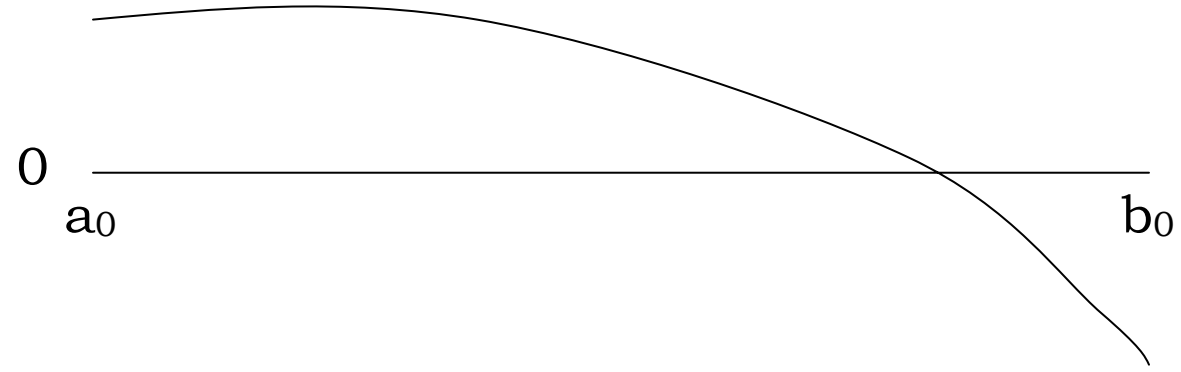
Usually easier to solve f'(x) = 0 than to deal with f(x) directly.

There are lots of ways to do this. Three popular approaches are bisection, functional iteration, and Newton-Raphson.

Bisection (for 1 dimension problems)

Have continuous function g(x) and two values a_0 and b_0 such that

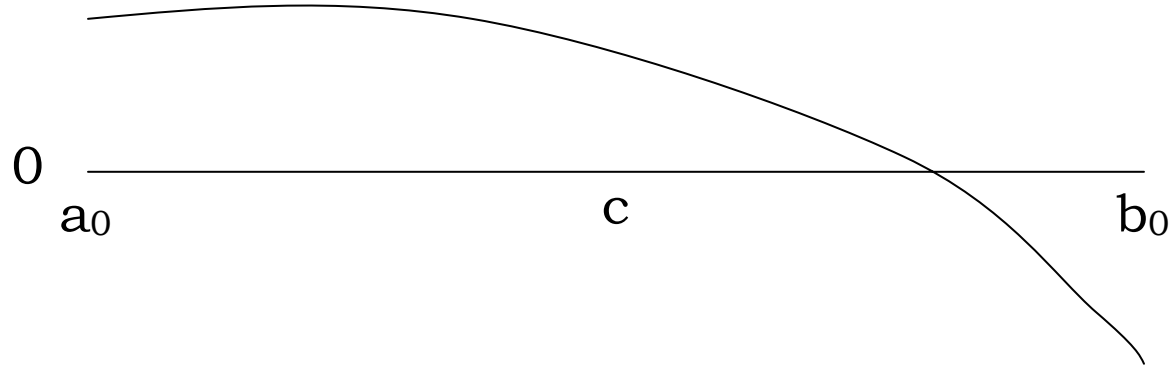
$g(a_0) > 0$ and $g(b_0) < 0$ (or vice versa)



We know that there exists at least one point, x^* in (a_0, b_0) such that $g(x^*) = 0$ by the intermediate value theorem.

Idea: try midpoint of interval

$$c = \frac{a_0 + b_0}{2} \text{ \& evaluate } g(c)$$



If $g(c)g(a_0) > 0$ set $a_1 = c$, $b_1 = b_0$ and continue

{g(c) and g(a) are both > 0 or both < 0 so must be a root between c and b}

If $g(c)g(a_0) < 0$ set $a_1 = a_0$, $b_1 = c$ and continue

{g(c) and g(a) are on opposite side so there must be a root between a and c}

If $g(c) = 0$ stop

Continue until the interval width gets small enough.

After n steps, the interval width = $b_n - a_n$

$= \frac{b_0 - a_0}{2^n}$.

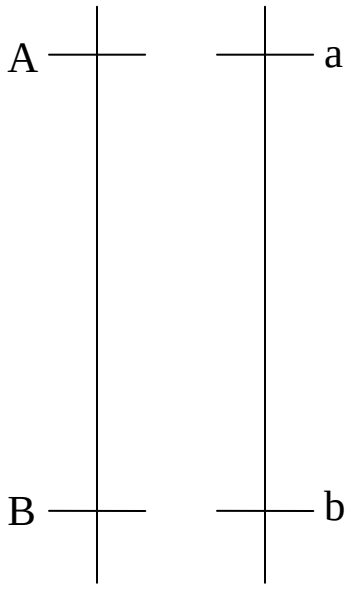
Set $\hat{x}^* = \frac{a_n + b_n}{2}$, the midpoint of the last interval as the estimate of the root $\hat{x}$.

Since it's the midpoint of the last interval, the maximum error satisfies

$$| \hat{x}^* - \hat{x} | \leq \frac{b_0 - a_0}{2^{n+1}}$$

Example: Linkage Analysis (Rao, 1973, pp 268-269)

2 gene on a chromosome are separated by a recombination fraction θ (θ ≤ ½).



This organism can pass 4 possible haplotypes to its offspring

Haplotype	Probability
AB	(1 – θ)/2
Ab	θ/2
aB	θ/2
ab	(1 – θ)/2

An experiment was performed to estimate θ. The breeding experiment crossed AB|ab x AB|ab and recorded the observed phenotypes.

In this experiment, 2 dominate traits were observed (A dominant to a, B dominate to b).

While there are 16 possible joint haplotypes in the offspring (4 from father times 4 from mother), there are only 4 possible phenotypes

Phenotype	Probability	Counts
AB	$(3 - 2\theta + \theta^2)/4$	125
Ab	$(2\theta - \theta^2)/4$	18
aB	$(2\theta - \theta^2)/4$	20
ab	$(1 - 2\theta + \theta^2)/4$	34

Note that this problem is easier to solve with the transformation (Lange page 126, problem 7)

$$\lambda = 1 - 2\theta + \theta^2 = (1 - \theta)^2$$

$$\theta = 1 - \sqrt{\lambda}$$

Under this transformation, the probabilities are

Phenotype	Probability	Counts
AB	$(2 + \lambda)/4$	125
Ab	$(1 - \lambda)/4$	18
aB	$(1 - \lambda)/4$	20
ab	$\lambda/4$	34

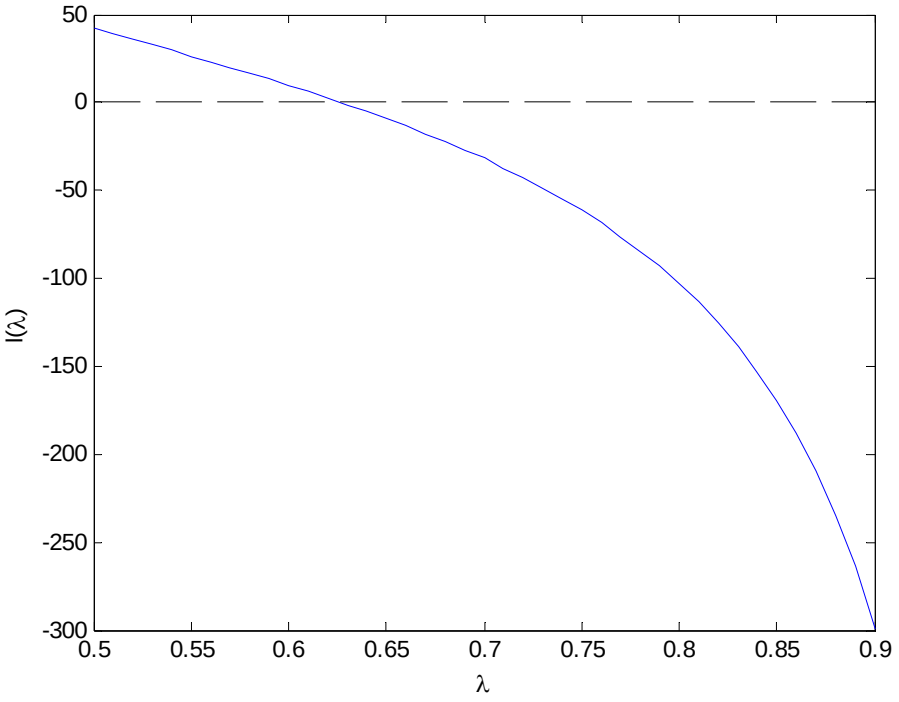
The likelihood and log likelihood functions are

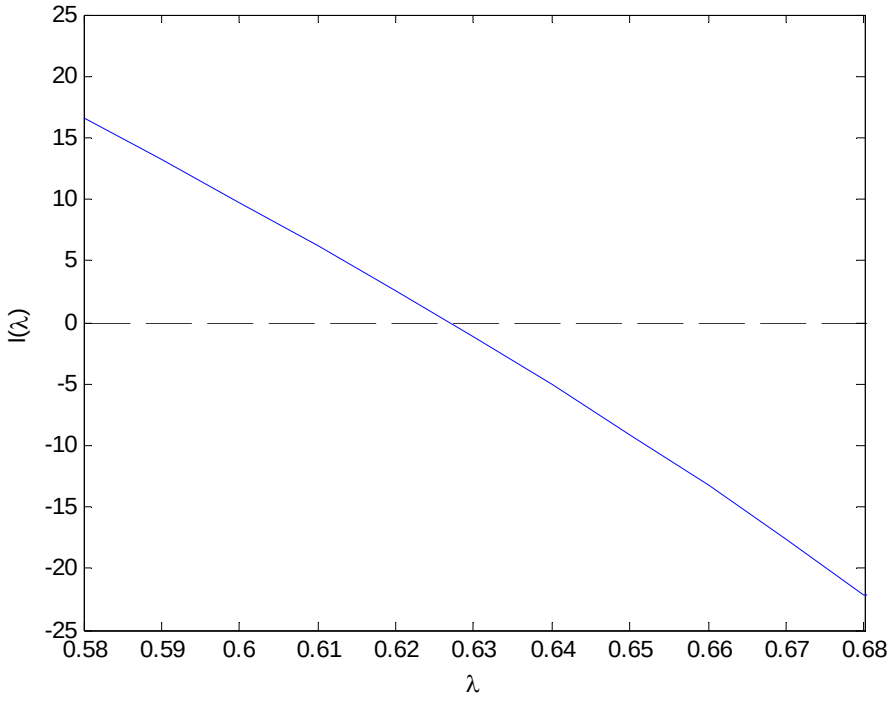
$$L(\lambda) \propto (2 + \lambda)^{125} (1 - \lambda)^{18+20} \lambda^{34}$$

$$\log L(\lambda) = 125 \log(2 + \lambda) + 38 \log(1 - \lambda) + 34 \log \lambda$$

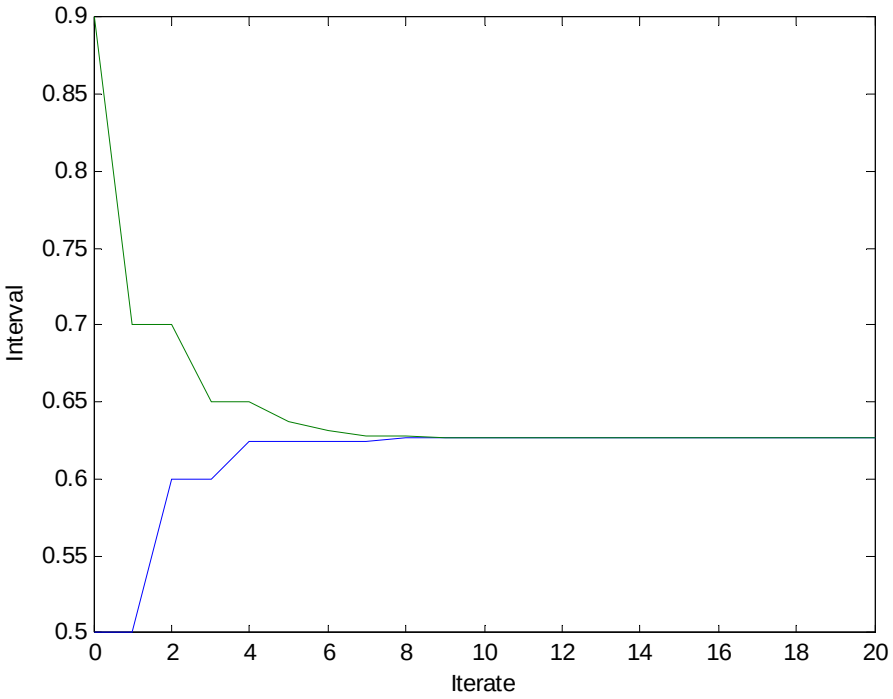
which gives the score function

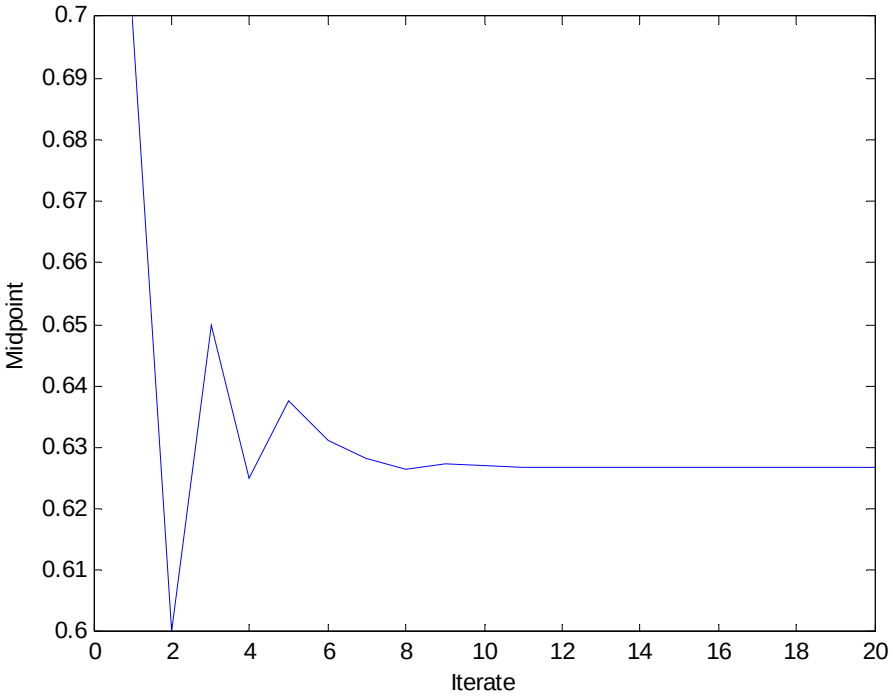
$$l(\lambda) = \frac{d}{d\lambda} \log L(\lambda) = \frac{125}{\lambda} - \frac{38}{1 - \lambda} + \frac{34}{\lambda}$$

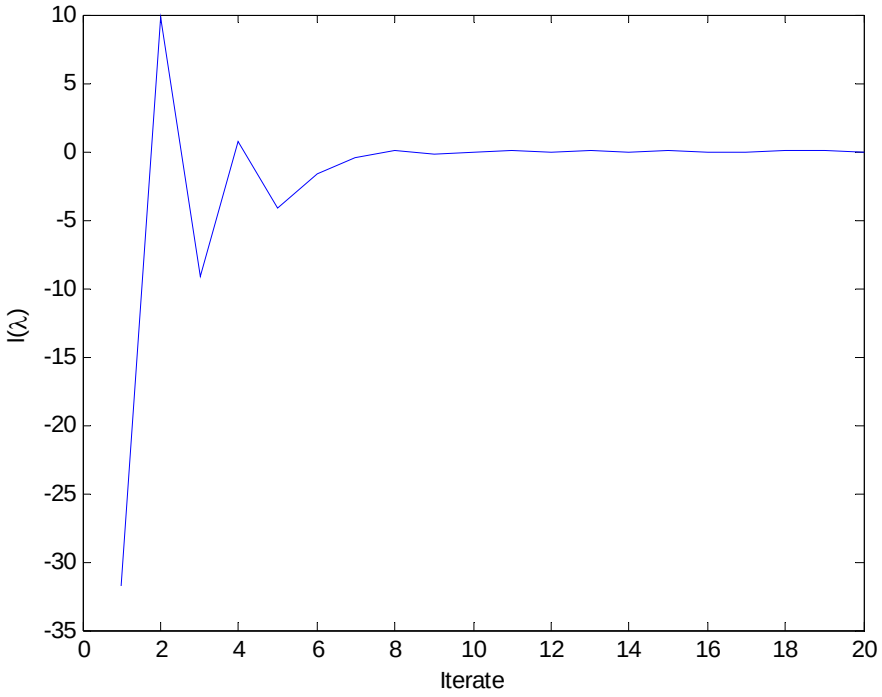

50
0
-50
-100
-150
-200
-250
-300
l(λ)
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
λ

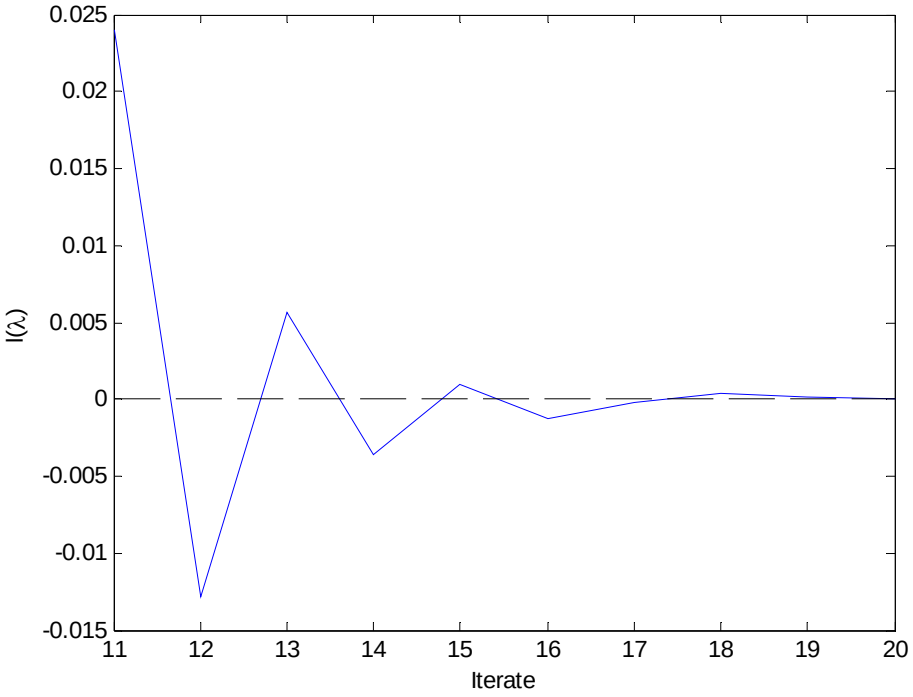

25
20
15
10
5
0
-5
-10
-15
-20
-25
l(λ)
0.58
0.59
0.6
0.61
0.62
0.63
0.64
0.65
0.66
0.67
0.68
λ

The bisection algorithm for $l(\lambda)$ with $a_0 = 0.5$ and $b_0 = 0.9$ gives $\hat{\lambda}^* = 0.6268$ after 20 steps. The convergence pattern can be seen with




0.7
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0.61
0.6
Midpoint
0
2
4
6
8
10
12
14
16
18
20
Iterate


10
5
0
-5
-10
-15
-20
-25
-30
-35
I(λ)
0
2
4
6
8
10
12
14
16
18
20
Iterate



Iterate	λ	l(λ)
1.0000	0.7000	-31.7989
2.0000	0.6000	9.7436
3.0000	0.6500	-9.0939
4.0000	0.6250	0.6857
5.0000	0.6375	-4.1009
6.0000	0.6312	-1.6835
7.0000	0.6281	-0.4931
8.0000	0.6266	0.0977
9.0000	0.6273	-0.1973
10.0000	0.6270	-0.0497
11.0000	0.6268	0.0240
12.0000	0.6269	-0.0128
13.0000	0.6268	0.0056
14.0000	0.6268	-0.0036
15.0000	0.6268	0.0010
16.0000	0.6268	-0.0013
17.0000	0.6268	-0.0002
18.0000	0.6268	0.0004
19.0000	0.6268	0.0001
20.0000	0.6268	-0.0000

For this example,

$$\text{max error} \le \frac{0.9 - 0.5}{2^{21}} = 1.9 \text{ x } 10^{-7}$$

To get $\hat{\theta}^*$, use $\theta = 1 - \sqrt{\lambda}$, which gives

$$\hat{\theta}^* = 1 - \sqrt{0.6268} = 0.2083$$

Note that the maximum error in with the estimate $\hat{\theta}^*$ needs to be carefully thought about, since the transformation is non-linear.

How many iterations for the bisection algorithm?

Once a_0 and b_0 are determined its easy. Base on a maximum desired error

Want $\frac{b_0 - a_0}{2^{n+1}} \le M$. Then set n to satisfy

$$n \ge \log_2\left(\frac{b_0 - a_0}{M}\right) - 1$$

$$= \log\left(\frac{b_0 - a_0}{M}\right) / \log 2 - 1$$

For example, for $M = 0.0001$

$$n \geq \log\left(\frac{0.9-0.5}{0.0001}\right)/\log 2 - 1 = 10.9$$

so use at least 11 iterates.

Advantages of the bisection method:

- Must terminate
- Guaranteed to find a zero of the function to desired accuracy

Disadvantages:

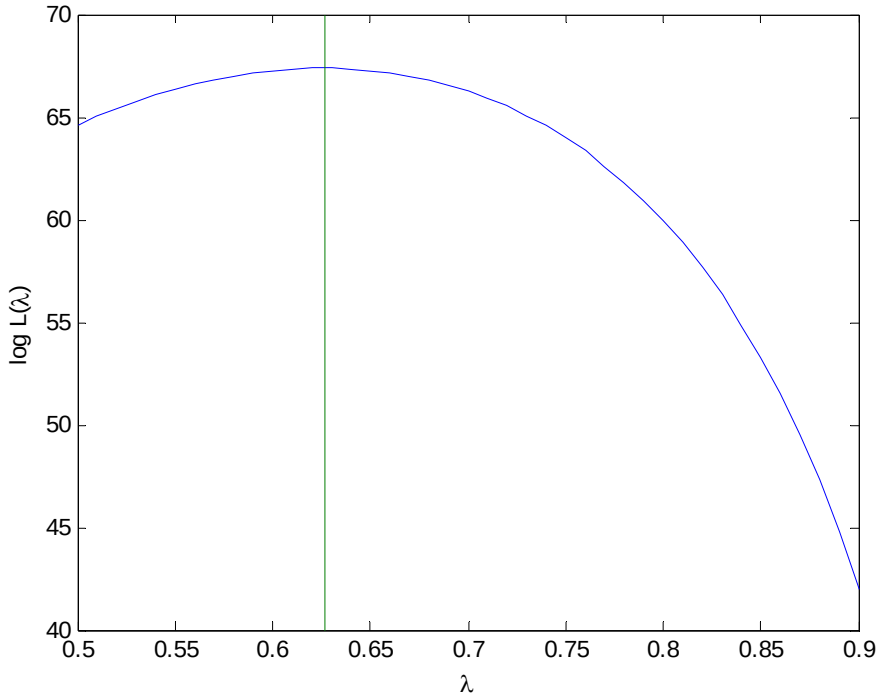
- Can only handle univariate problems
- Linear convergence (Other algorithms, such as Newton-Raphson can be faster)
- From optimization point of view, not guaranteed to find an optima.

Note that this disadvantage is not really specific to bisection, but to using root finders on the derivative of the function to be optimized.

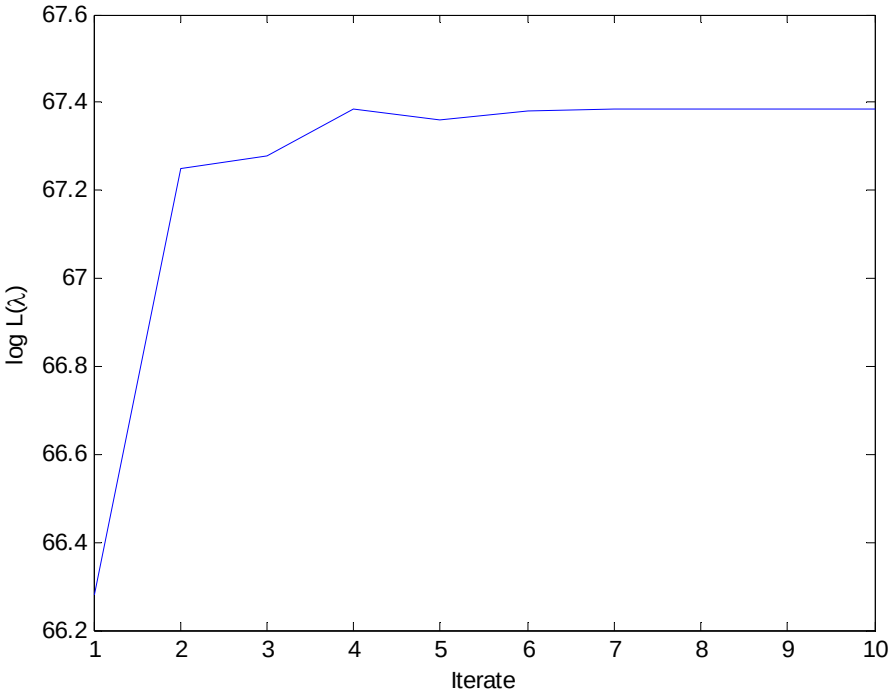
Solving $f'(x) = 0$ may give a minimum or a saddle point when a maximum is desired.

Need to check $f(x)$ or $f''(x)$ to see if $\hat{x}^*$ is a local maximum (e.g. is $f''(\hat{x}^*) < 0$ or is $f'(x)$ a decreasing function around $\hat{x}^*$)

In this case, the log likelihood is definitely concave, so we have found the MLE



One thing to note with the bisection algorithm when used to optimize a likelihood function, the likelihood (or log likelihood) does not have to increase at each step, particularly for the early iterates. However it will tend to do this once you are in the area of the optima, as can be seen in the following figure.


67.6
67.4
67.2
67
66.8
66.6
66.4
66.2
log L(λ)
1
2
3
4
5
6
7
8
9
10
Iterate

For comparison, the true MLE can be calculated for the linkage example. As seen last time

$$l(\lambda) = \frac{125}{2+\lambda} - \frac{38}{1-\lambda} + \frac{34}{\lambda} = 0$$

which is equivalent to solving

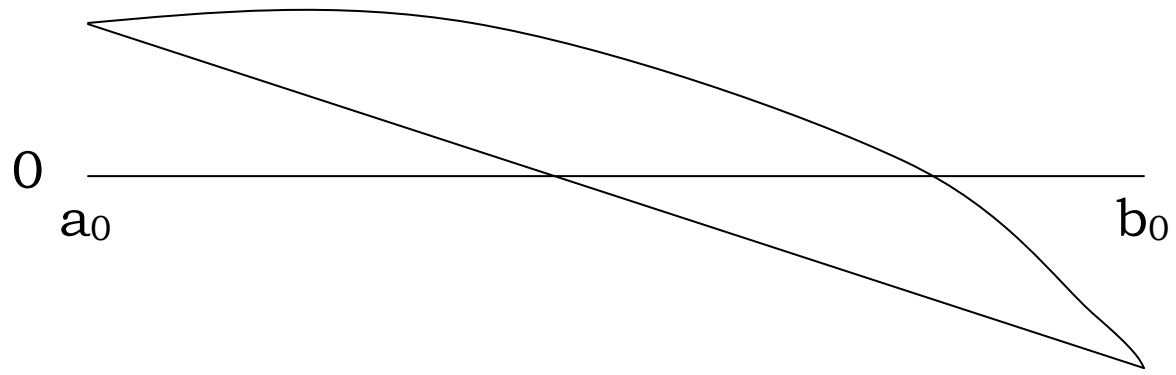
$$125(1 - \lambda)\lambda - 38(2 + \lambda)\lambda + 34(2 + \lambda)(1 - \lambda) = 0$$

$$= -197\lambda^2 + 15\lambda + 68$$

The two roots of this equation are 0.6268215 and -0.5506794. Only the first one is valid since λ must be in the range [0.25, 1].

There are other approaches similar to bisection. One useful one is the method of False Position (Regula Falsi).

A motivation behind this method is that the function is approximately linear in the region of interest.



Join points $(a_i, g(a_i))$ and $(b_i, g(b_i))$ with a straight line and find the point where the straight line intersects with the line y = 0 (call point p_i.

$$\text{Line: } l(x) = g(a_i) + \frac{g(b_1) - g(a_i)}{b_i - a_i}(x - a_i)$$

$$l(x) = 0 \Rightarrow p_i = a_i - \frac{(b_i - a_i)g(a_i)}{g(b_1) - g(a_i)}$$

If $g(p_i)g(a_i) > 0$ set $a_{i+1} = p_i$ and $b_{i+1} = b_i$

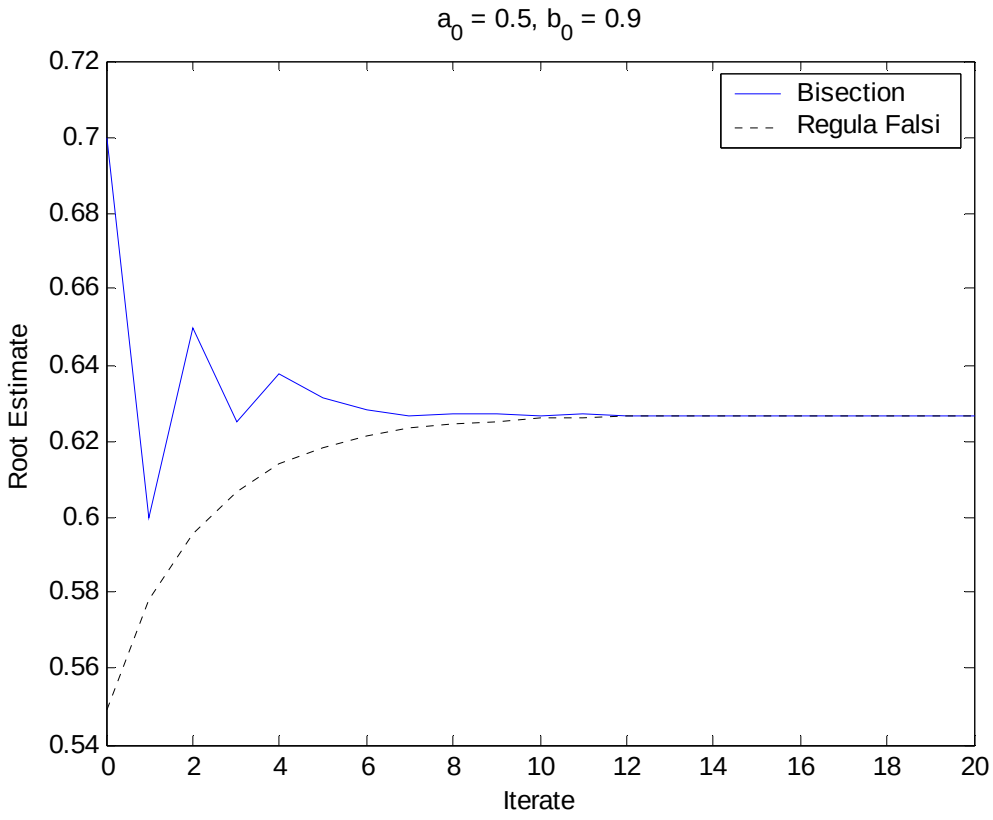
If $g(p_i)g(a_i) < 0$ set $a_{i+1} = a_i$ and $b_{i+1} = p_i$

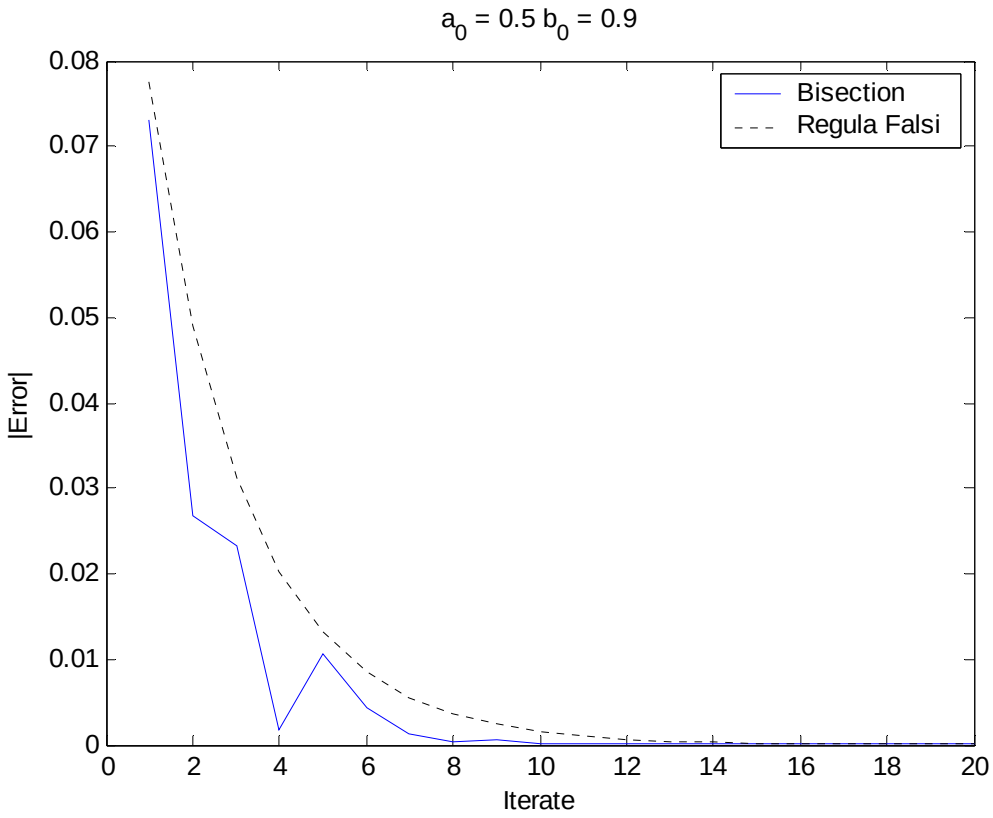
For some problems, this approach can be faster than bisection, but it depends on the shape of the function and the starting endpoints.

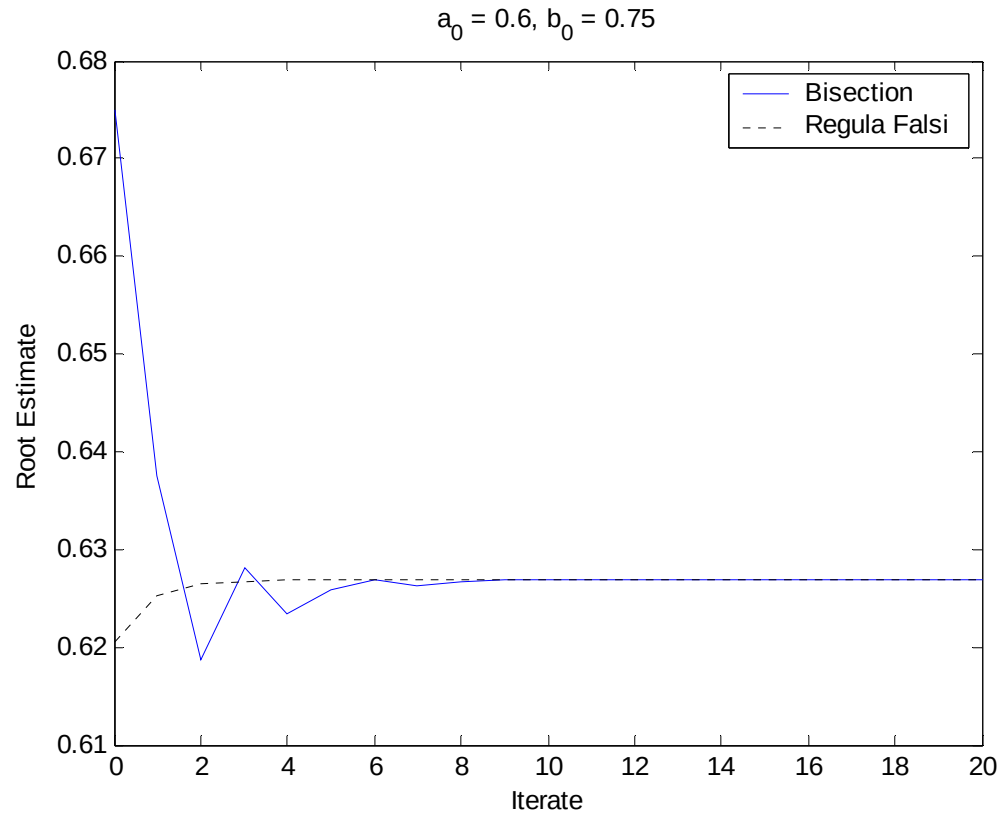
Also its harder to show convergence to the root since the interval size doesn't have to go to zero like with bisection. The can happen with a convex or concave function.

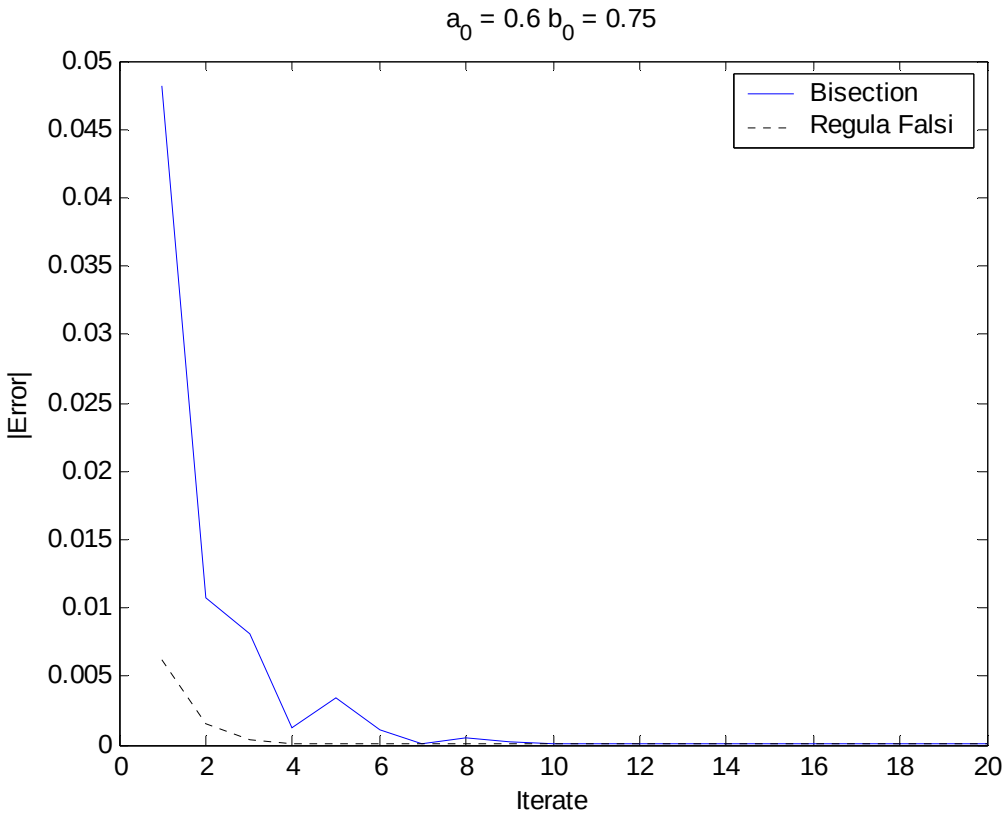
However, this routine will eventually converge to a root. This can be shown since $\{a_i\}$ is an non-decreasing sequence bounded above and $\{b_i\}$ is an non-increasing sequence bounded below.

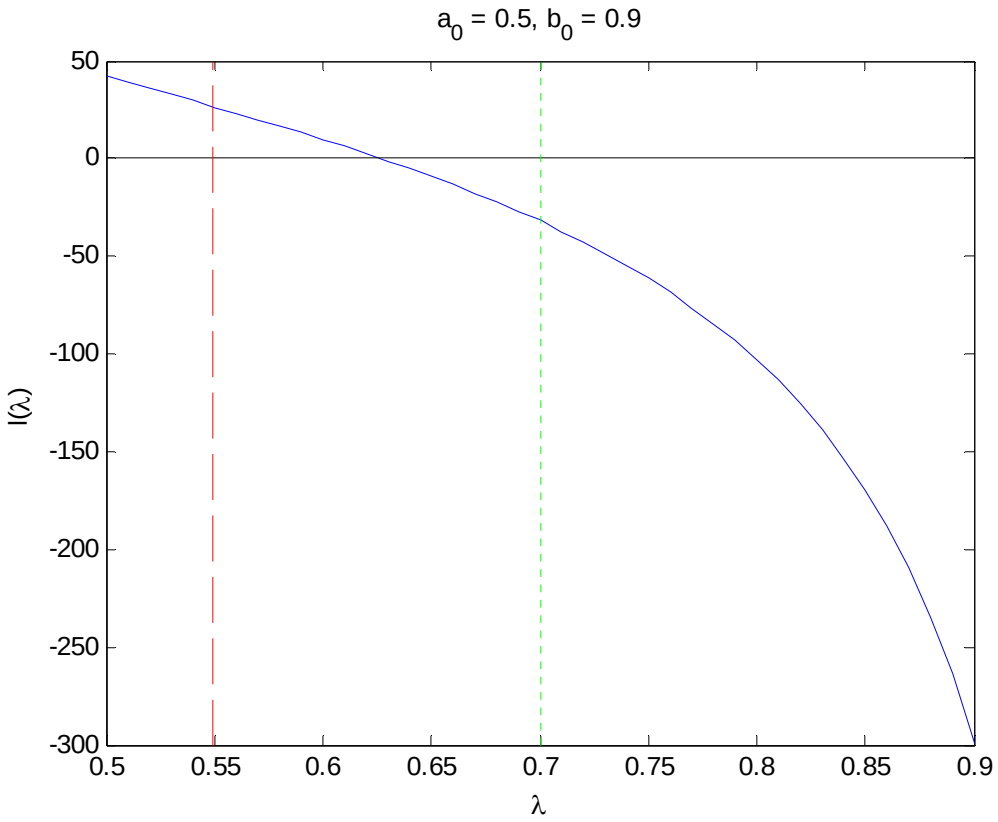
	Bisection		Regula Falsi	
Iterate	Lower	Upper	Lower	Upper
0	0.5000	0.9000	0.5000	0.9000
1	0.5000	0.7000	0.5492	0.9000
2	0.6000	0.7000	0.5779	0.9000
3	0.6000	0.6500	0.5955	0.9000
4	0.6250	0.6500	0.6066	0.9000
5	0.6250	0.6375	0.6137	0.9000
6	0.6250	0.6312	0.6183	0.9000
7	0.6250	0.6281	0.6212	0.9000
8	0.6266	0.6281	0.6232	0.9000
9	0.6266	0.6273	0.6244	0.9000
10	0.6266	0.6270	0.6253	0.9000
11	0.6268	0.6270	0.6258	0.9000
12	0.6268	0.6269	0.6262	0.9000
13	0.6268	0.6269	0.6264	0.9000
14	0.6268	0.6268	0.6265	0.9000
15	0.6268	0.6268	0.6266	0.9000
16	0.6268	0.6268	0.6267	0.9000
17	0.6268	0.6268	0.6267	0.9000
18	0.6268	0.6268	0.6268	0.9000
19	0.6268	0.6268	0.6268	0.9000
20	0.6268	0.6268	0.6268	0.9000

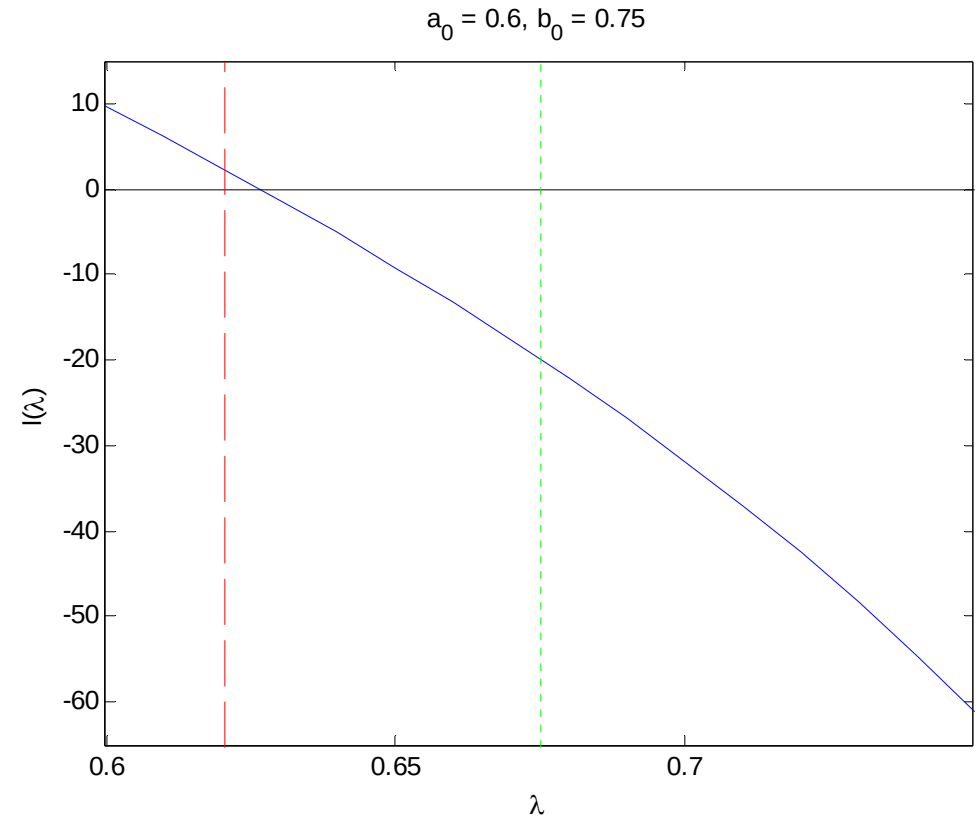

a_0 = 0.5, b_0 = 0.9
Bisection
Regula Falsi
Root Estimate
Iterate
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
0.54
0
2
4
6
8
10
12
14
16
18
20


a_0 = 0.5 b_0 = 0.9
Bisection
Regula Falsi
|Error|
Iterate
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
2
4
6
8
10
12
14
16
18
20


a_0 = 0.6, b_0 = 0.75
Bisection
Regula Falsi
Root Estimate
Iterate
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0.61
0
2
4
6
8
10
12
14
16
18
20


a_0 = 0.6 b_0 = 0.75
Bisection
Regula Falsi
|Error|
Iterate
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
2
4
6
8
10
12
14
16
18
20


a_0 = 0.5, b_0 = 0.9
50
0
-50
-100
-150
-200
-250
-300
l(λ)
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
λ


a_0 = 0.6, b_0 = 0.75
10
0
-10
-20
-30
-40
-50
-60
l(λ)
0.6
0.65
0.7
λ

Functional Iteration (Fixed Point Approaches)

Instead of solving g(x) = 0, we can investigate the function

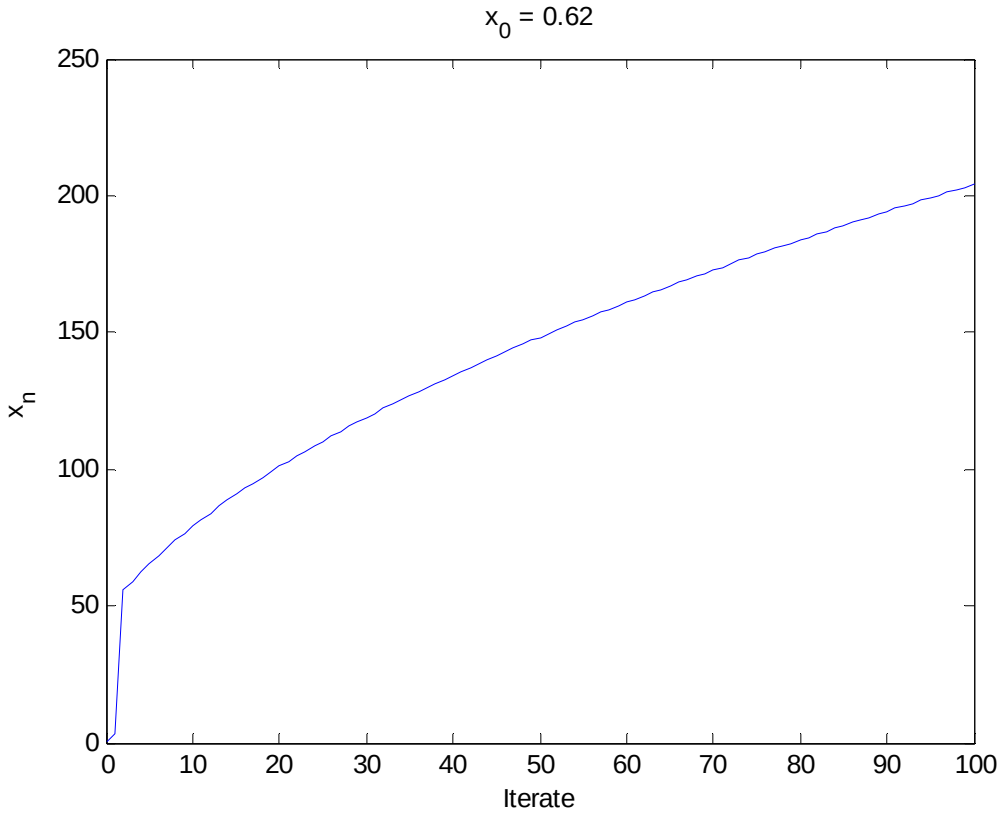
$$f(x) = g(x) + x$$

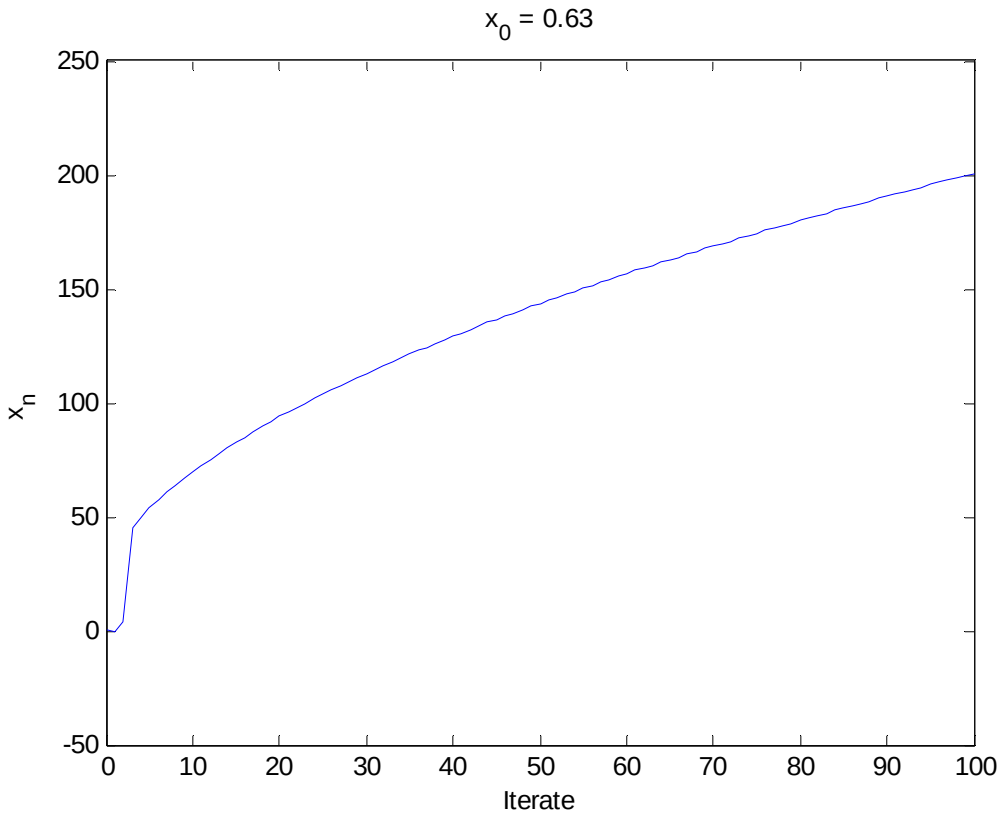
Solving g(x) = 0 is the same as solving f(x) = x.

In many situations, iterates of the sequence $x_n = f(x_{n-1})$ converge to a root of g(x) starting from any point x_0 nearby.

But it doesn't have to!

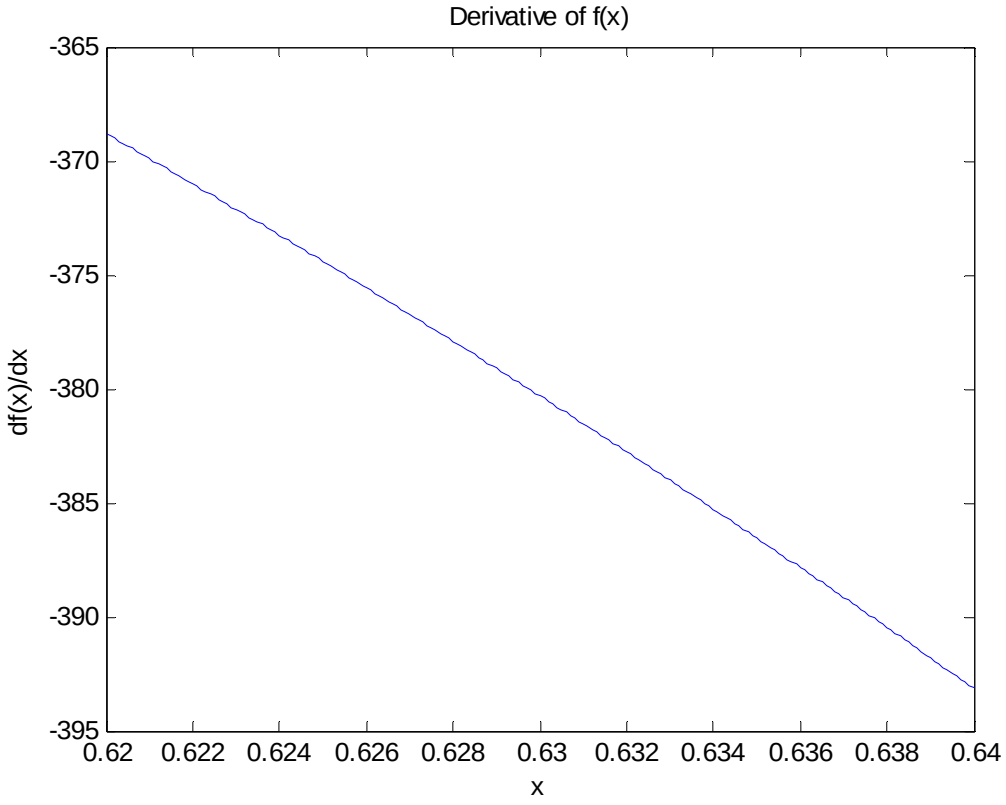
Lets run this algorithm starting at $x_0 = 0.62$ and $x_0 = 0.63$, which are both close to the true root of 0.6268215.


x_0 = 0.62
250
200
150
100
50
0
x_n
0
10
20
30
40
50
60
70
80
90
100
Iterate


x_0 = 0.63
250
200
150
100
50
0
-50
x_n
0
10
20
30
40
50
60
70
80
90
100
Iterate

In both cases, the iterates seem to diverge, or at least don't seem to converge in the right region.

Lets look at the function f(x), in particular its derivative.



So small changes in x lead to large changes in f(x), even very close to the fixed point.

So we need conditions on when fixed point methods can work

Proposition 5.3.1: Suppose f(x) defined on a closed interval I satisfies the conditions

1) $f(x) \in I$ whenever $x \in I$

2) $|f(y) - f(x)| \leq \lambda |y - x|$ for any two points x & y in I.

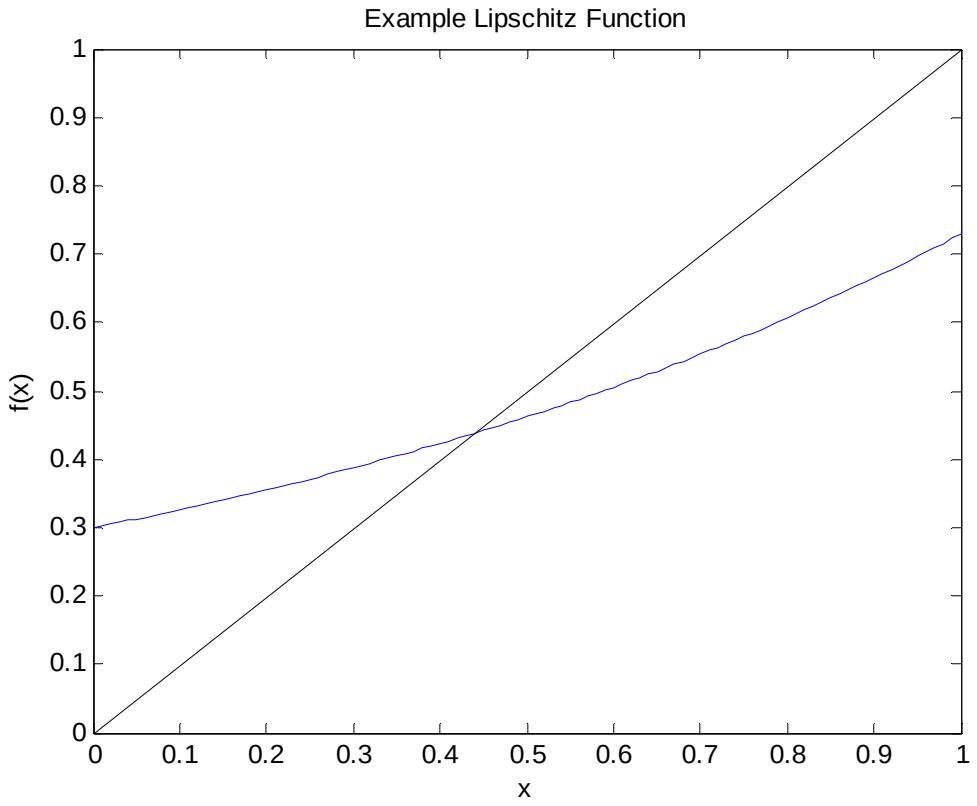
Then provided the Lipschitz constant λ is in [0, 1), f(x) has a unique fixed point $x_\infty \in I$, and the functional iterates $x_n = f(x_{n-1})$ converge to x_∞ regardless of the starting point $x_0 \in I$. Furthermore, we have the precise error estimate

$$|x_n - x_\infty| \leq \frac{\lambda^n}{1-\lambda} |x_1 - x_0|$$

(For proof, see Lange)

So if f(x) (and g(x)) is nice enough, we can be guaranteed to find the desired root.

Need to get a handle on the Lipschitz constant λ. Usually you can use an upper bound on $|f'(x)|$.



What happened in the linkage example. Well $\lambda > 360$. So for an x near (but not equal to x_∞)

$|f(x) - x_\infty| = |f'(z)(x - x_\infty)|$ (Mean value theorem)

$$\geq 360\ |x - x_\infty| \geq |x - x_\infty|$$

So there was no way that this approach could work. This situation is known as repulsive. You have to end up further from the fixed point than where you started.

If $|f'(x_\infty)| < 1$, the situation is known as attractive.

If $|f'(x_\infty)| = 1$, the situation is indeterminant and investigation of the function is required.

Example: Extinction Probabilities of Branching Processes (Section 5.3.2)

Stochastic process that describes a model of population growth.

Start with 1 particle. This particle has k offspring with probability p_k. Each of these k particles generates offspring by the same mechanism. And so on for these offspring.

One question of interest is whether the population will completely die out.

This question can be answered by investigating the generating function of the process

$$P(s) = \sum_{k=0}^{\infty} p_k s^k$$

If $p_0 = 0$, the population can never die out, so we will only consider the case $p_0 > 0$.

It ends up that the probability that the population will eventually die out satisfies the fixed point equation

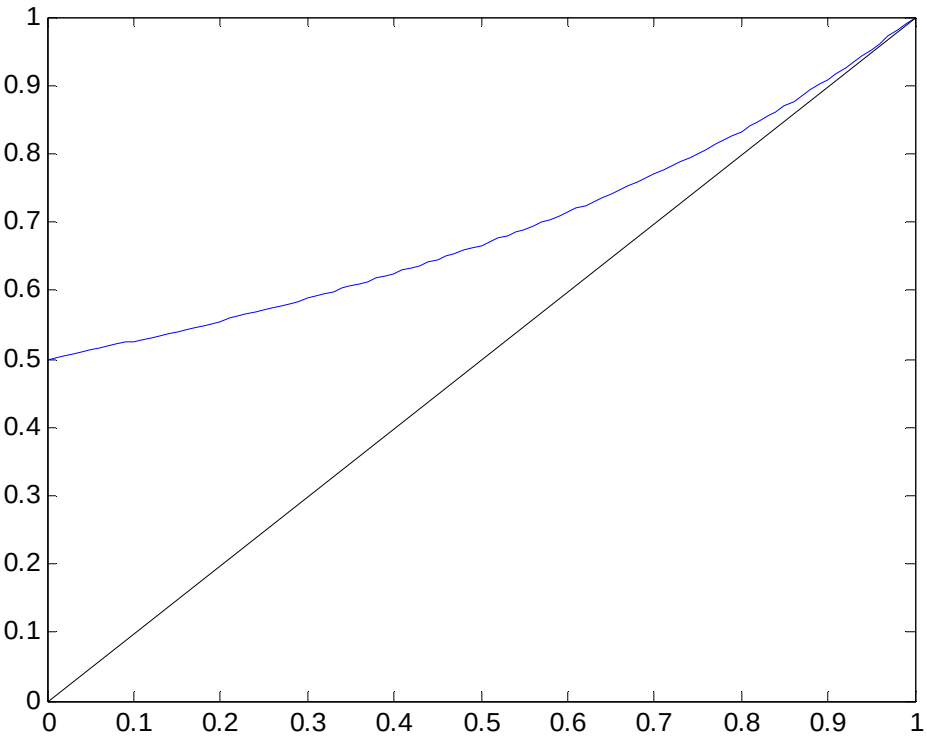
$$s = P(s)$$

This equation can have 1 or 2 fixed points. The point s = 1 must be one as $\sum_{k=0}^{\infty} p_k = 1$. It can be two since P(0) > 0 (we'll also ignore the case where $p_0 = 1$) and P(s) is a convex function in [0,1] since

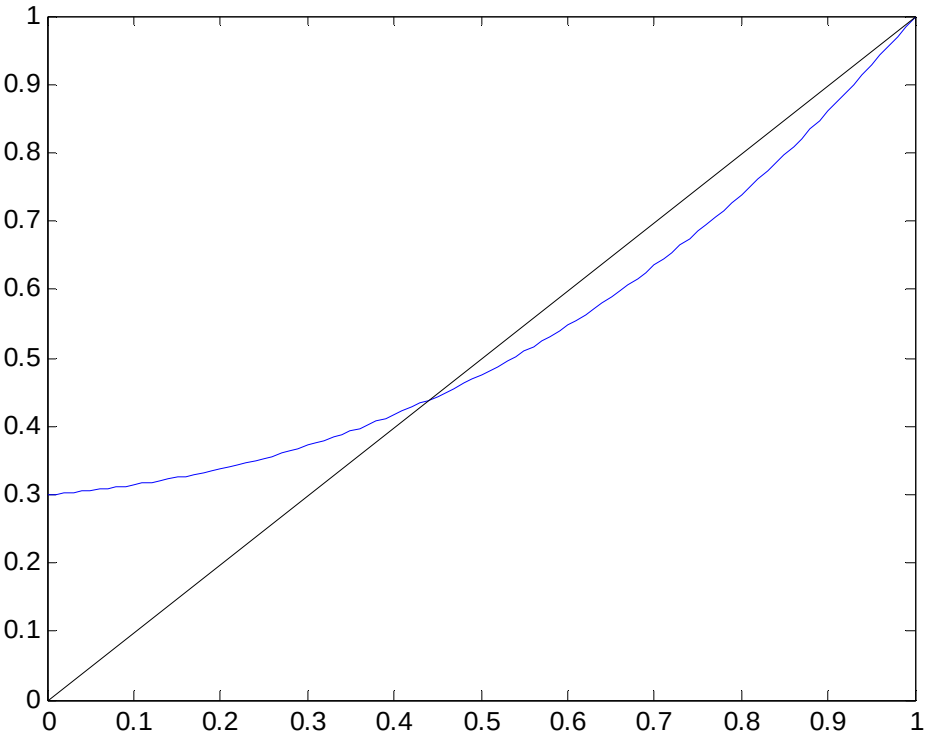
$$P''(s) = \sum_{k=2}^{\infty} k(k-1)p_k s^{k-2} > 0$$

Since its convex, it will intersect a straight line at most twice. However the second point of intersection may not be in [0,1] (if it exists). The function must look like one of the following.

P(s) = 1 / (2 - s); $p_k = 2^{-k-1}$



P(s) = $0.3 + 0.1s + 0.4s^2 + 0.2s^3$



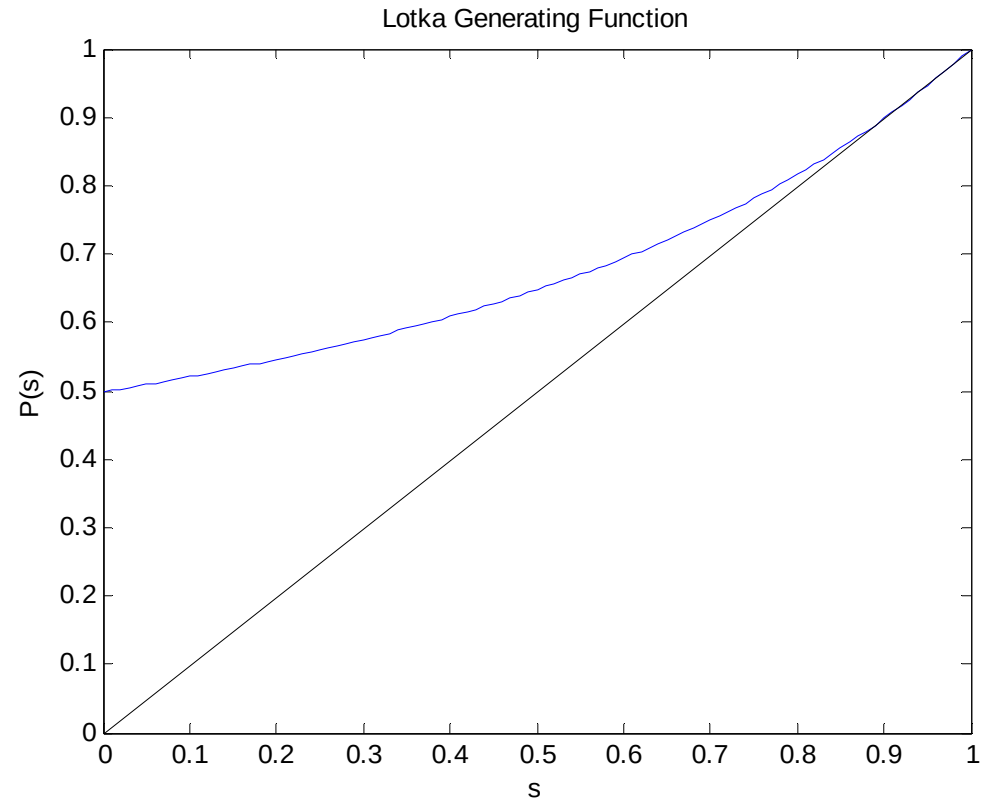
Which occurs depends on P'(1), the mean number of offspring for each particle. If $P'(1) \leq 1$, the first situation must happen. The second situation will occur with If $P'(1) > 1$.

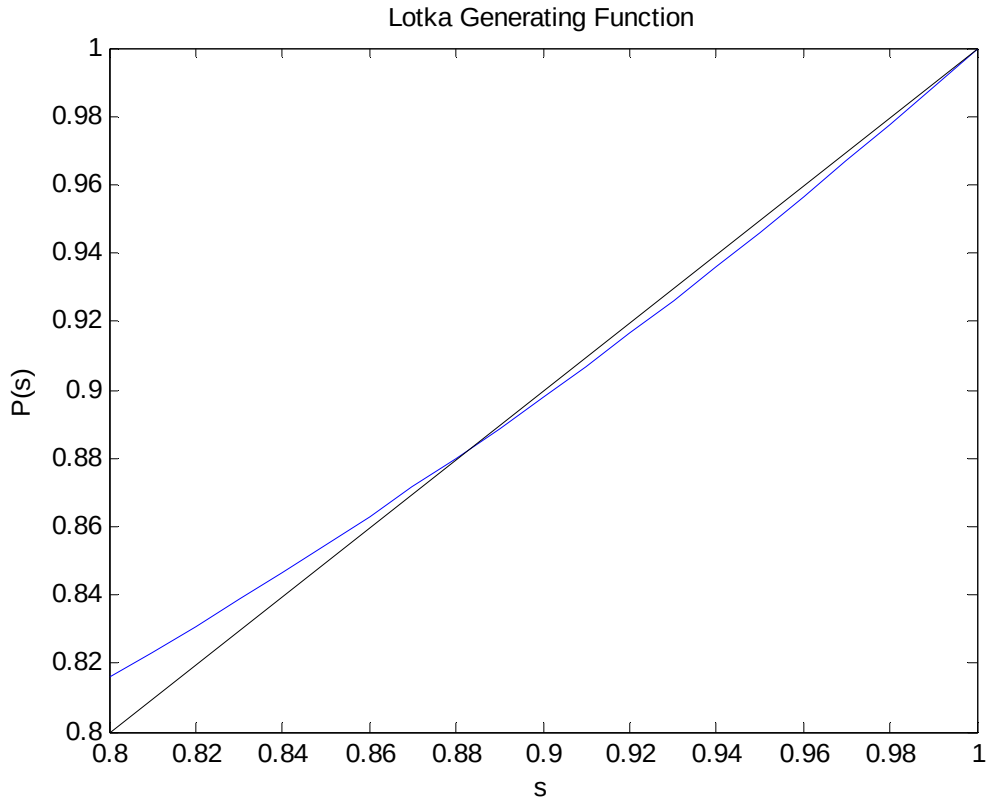
Note that the extinction probability is the smaller fixed point when $P'(1) > 1$. ($s = 1$ is a point of repulsion).

When $P'(1) > 1$, we can find the fixed point by iterating starting at $s_0 = 0$. This works since $0 < P'(s) < 1$ and $P(s) \leq s$ for $s \in [0, s_\infty]$. Usually it will be for $s \in [0, s_\infty + \delta]$, where $\delta > 0$.

Lets look at Lotka's example examining the extinction of surnames among white male in the US based on 1920 census data.

$$P(s) = 0.4982 + 0.2103s + 0.1270s^2 + 0.7330s^3 + 0.0418s^4 + 0.0241s^5 + 0.0132s^6 + 0.0069s^7 + 0.0035s^8 + 0.0015s^9 + 0.0005s^{10}$$


Lotka Generating Function
P(s)
s
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

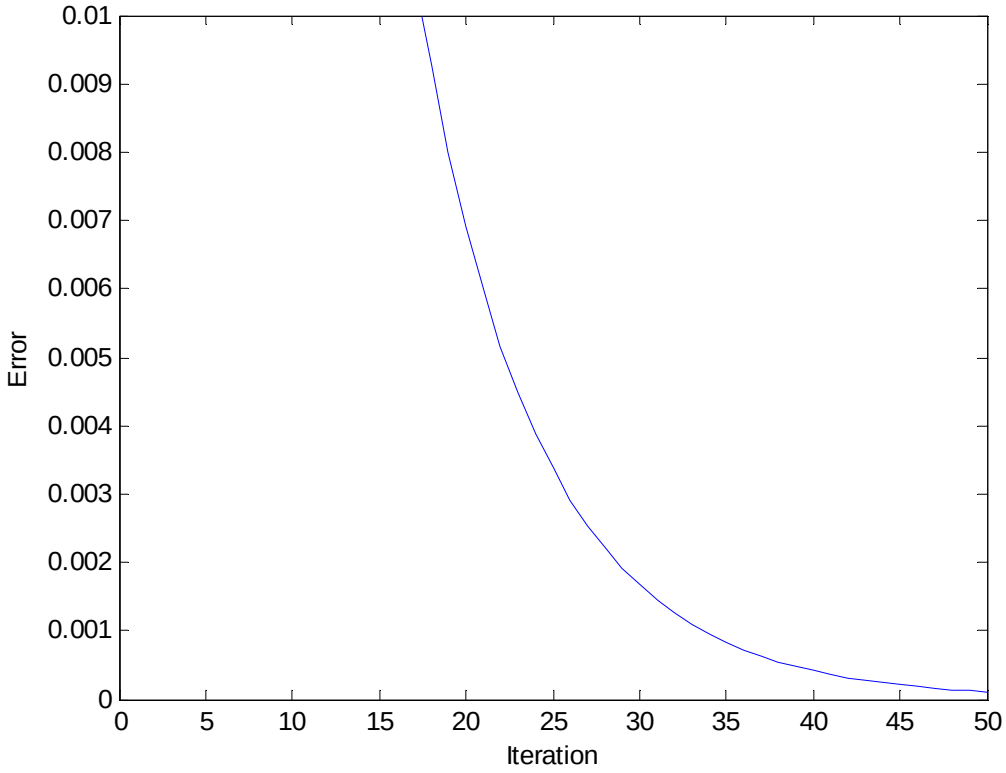

Lotka Generating Function
P(s)
s
0.8
0.82
0.84
0.86
0.88
0.9
0.92
0.94
0.96
0.98
1

The method converges slowly to the extinction probability of 0.879755.

After 50 steps, we can show that

Bound: $|s_n - s_\infty| \le \frac{\lambda^n}{1-\lambda} |s_1 - s_0| = 0.0039$

Actual: $|s_n - s_\infty| = 0.000105$



Why is convergence slow?

Error estimate bound

$$|s_n - s_\infty| \le \frac{\lambda^n}{1-\lambda} |s_1 - s_0|$$

So

$$\frac{\max|s_{n+1} - s_\infty|}{\max|s_n - s_\infty|} = \lambda$$

A reasonable Lipschitz constant for this problem is $P'(s_\infty) = 0.8713$.

So each step is only bringing you about 13% closer to the truth.

If we were to use the bisection method to solve this problem, based on $g(s) = P(s) - s = 0$.

$$\frac{\max|s_{n+1} - s_\infty|}{\max|s_n - s_\infty|} = \tfrac{1}{2}$$

So each step is bringing us about half the way there.

After 50 bisection steps ($a_0 = 0$, $b_0 = 1$)

$$|s_n - s_\infty| \le \frac{1}{2^{51}} = 4.4409\text{e-}016$$

Functional iteration methods such as these aren't commonly used to directly find roots much in statistics from what I've seen, but other methods, such as Newton-Raphson to have a functional iteration property underlying them.

Newton – Raphson

Probably the most popular root finding method.

Based on Taylor series approximation

$$g(x_{n-1}) = g(x_{n-1}) - g(x_\infty)$$
$$= g'(z)(x_{n-1} - x_\infty)$$

where z between x_{n-1} and x_∞. If we plug x_n in place of x_∞, we get the following updating equation

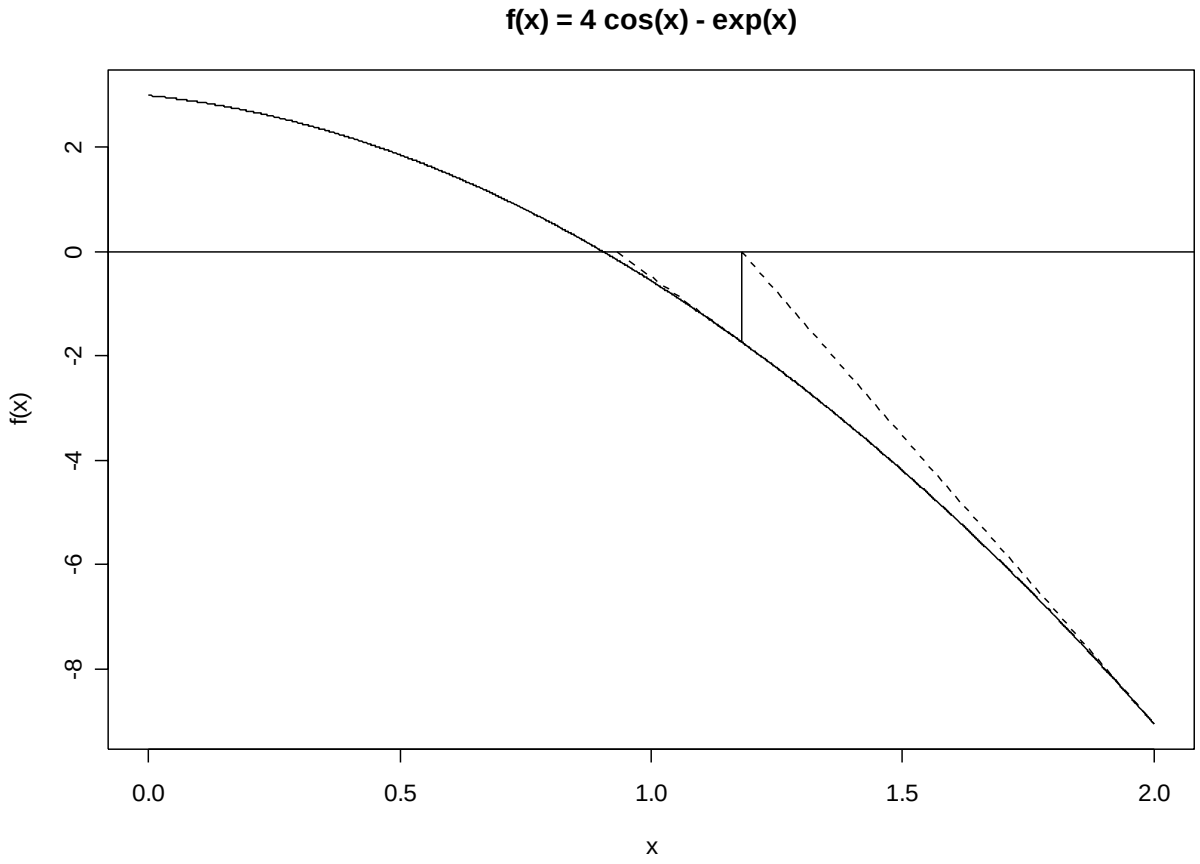
$$x_n = x_{n-1} - \frac{g(x_{n-1})}{g'(x_{n-1})}$$

Geometric Interpretation of updating formula

Finds tangent line to curve at $(x_{n-1}, g(x_{n-1}))$

$$l(x) = g(x_{n-1}) + g'(x_{n-1})(x - x_{n-1})$$

and solves $l(x) = 0$ to give x_n. This sequence is continued until convergence.



Example: Variance Heterogeneity

$$Y_i \,|\, x_i \sim N(\mu, \mu^2 x_i^2); \quad i = 1, \ldots, n$$

Since the variance depends on the mean, the $\bar{y}$ will not be the MLE in this case.

$$L(\mu) \propto \prod_{i=1}^{n} \frac{1}{\mu x_i} \exp\left(-\frac{1}{2} \left(\frac{y_i - \mu}{\mu x_i} \right)^2 \right)$$

$$\log L(\mu) = -n \log \mu - \frac{1}{2} \sum_{i=1}^{n} \left(\frac{y_i - \mu}{\mu x_i} \right)^2 + c(\mathbf{x})$$

$$l(\mu) = \frac{d \log L(\mu)}{d\mu} = -\frac{n}{\mu} + \frac{1}{\mu^3} \sum_{i=1}^{n} \left(\frac{y_i - \mu}{x_i} \right) \frac{y_i}{x_i}$$

$$l'(\mu) = \frac{d^2 \log L(\mu)}{d\mu^2}$$

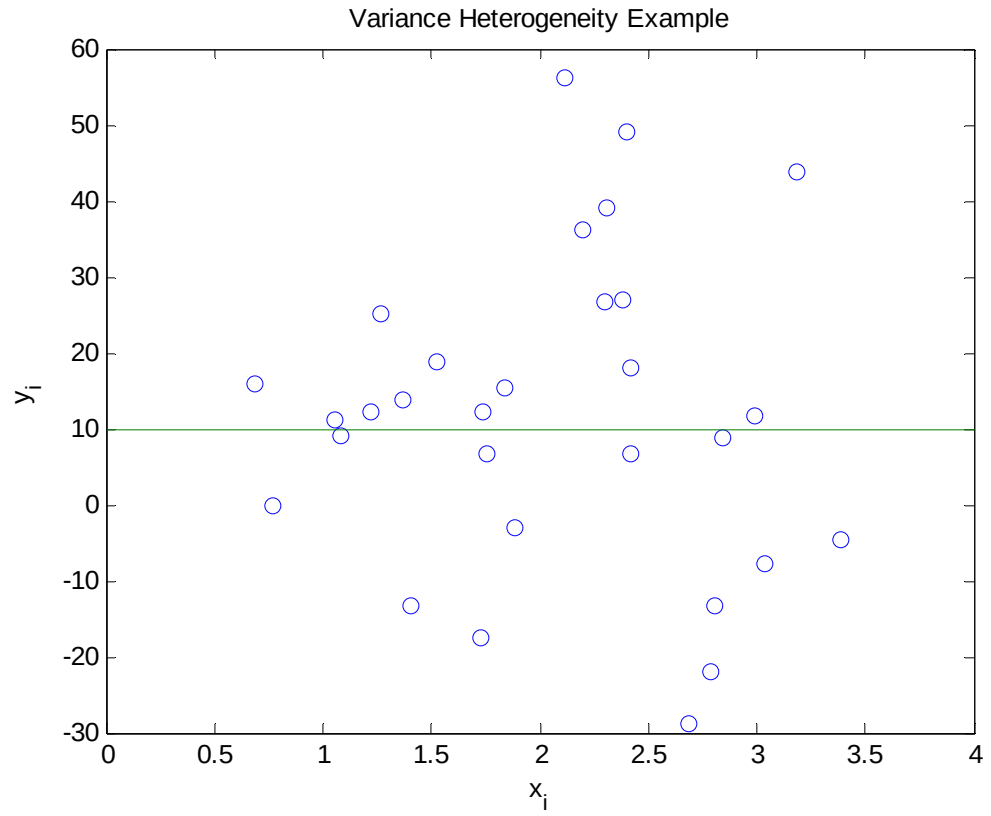
$$= \frac{n}{\mu^2} - \frac{3}{\mu^4} \sum_{i=1}^{n} \left(\frac{y_i - \mu}{x_i} \right) \frac{y_i}{x_i} - \frac{1}{\mu^3} \sum_{i=1}^{n} \frac{y_i}{x_i^2}$$

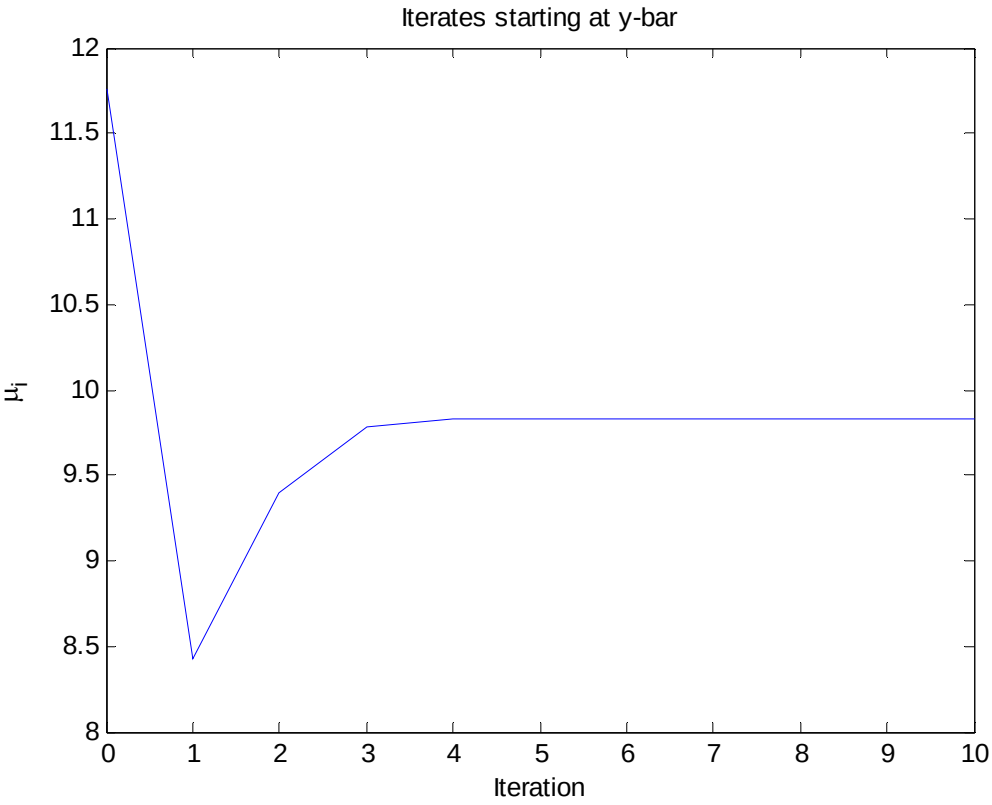
The Newton scheme for this problem is given by

$$\mu_n = \mu_{n-1} - \frac{l(\mu_{n-1})}{l'(\mu_{n-1})}$$

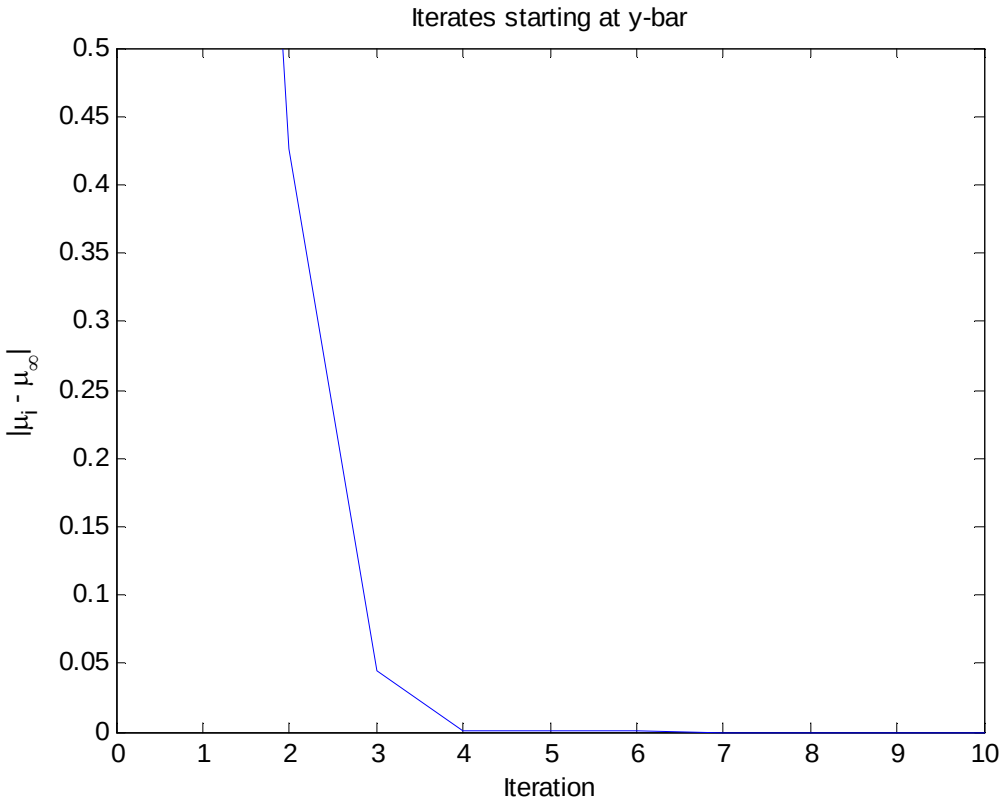
As an example, 30 independent observations were generated in Matlab from $Y_i \,|\, x_i \sim N(10, 10^2 x_i^2)$ where $X_i \sim \sqrt{\chi_4^2}$.

For a starting point, $\mu_0 = \bar{y} = 11.7646$, a method of moment estimator will be used.


Variance Heterogeneity Example
60
50
40
30
20
10
0
-10
-20
-30
0
0.5
1
1.5
2
2.5
3
3.5
4
y_i
x_i


Iterates starting at y-bar
12
11.5
11
10.5
10
9.5
9
8.5
8
0
1
2
3
4
5
6
7
8
9
10
μ_i
Iteration

As can be seen, the Newton-Raphson scheme converges quickly the the MLE $\hat{\mu}$ = 9.8279. Note that this is quite a bit lower than $\bar{y}$ = 11.7646



Iteration	μ_i	$\mu_i - \mu_\infty$
0	11.7646	1.9367
1.0000	8.4280	-1.3999
2.0000	9.4024	-0.4255
3.0000	9.7830	-0.0449
4.0000	9.8274	-0.0005
5.0000	9.8279	-0.0000
6.0000	9.8279	-0.0000
7.0000	9.8279	0

Instead of starting at $\bar{y}$, lets start at μ_0 = 15.1. The sequence of iterates quickly diverges.

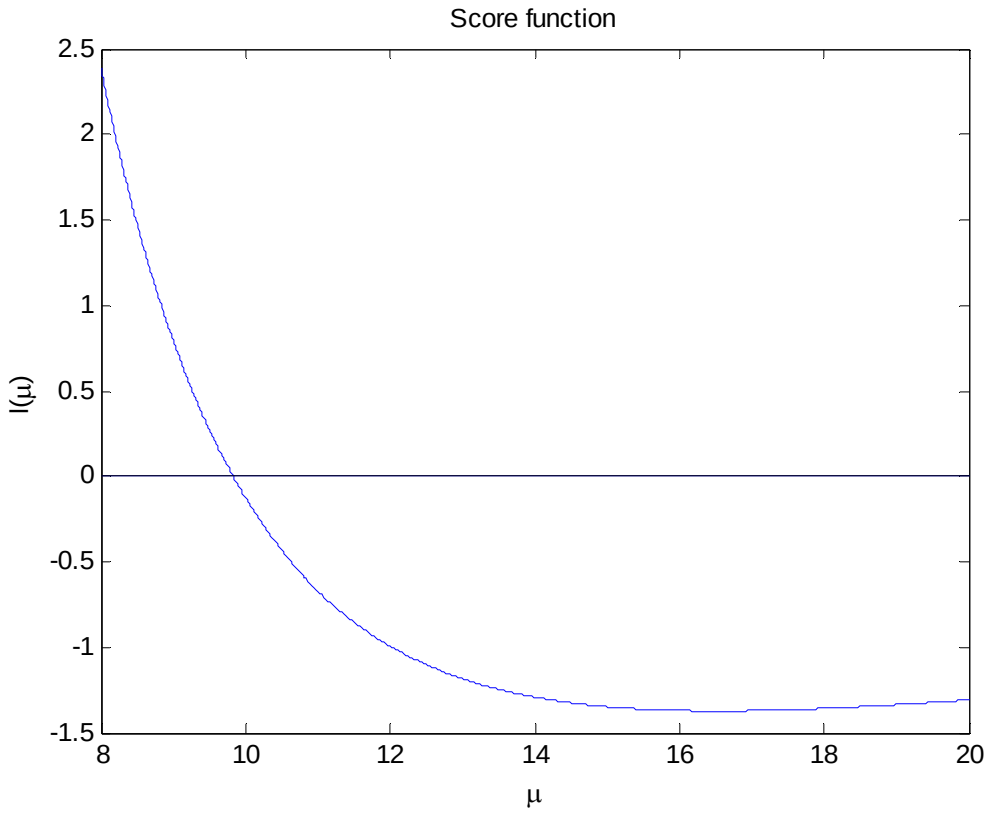
Iteration	μ_i
0	15.1000
1	-24.8822
2	335.4398
3	667.3233

However if we start close by at μ_0 = 15, we converge to where we want. Note however that we do take a weird path.

Iteration	μ_i
0	15.0000
1	-21.4034
2	9.1677
3	9.7235
4	9.8251
5	9.8279

In fact, the Newton-Raphson scheme will converge to 9.8279 if $\mu_0 \in (0, 15.01026)$. If $\mu_0 > 15.01026$, the procedure appears to diverge to ∞.

So the starting point matters. Lets look at the score function $l(\mu)$.



So when μ gets around 14 or 15, the function gets very flat ($l'(\mu)$ is close to zero), so the first iteration takes the sequence far from the zero.

In fact when μ is greater than 18, the zero, assuming that there is one (probably isn't), is in the other direction from what we want.

These results can be seen from the updating formula

$$\mu_n = \mu_{n-1} - \frac{l(\mu_{n-1})}{l'(\mu_{n-1})}$$

Convergence of Newton-Raphson

Note that updating formula is of the functional iteration form

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

so we can use the methods earlier to investigate the convergence properties of Newton – Raphson.

As seen last time, the convergence depends on $f'(x)$. For Newton

$$f'(x) = 1 - \frac{g'(x)}{g'(x)} + \frac{g(x)g''(x)}{g'(x)^2} = \frac{g(x)g''(x)}{g'(x)^2}$$

As seen last time, we need $-1 < f'(x) < 1$ for the sequence to converge to a fixed point. Notice that the above depends of $g'(x)$, which is the derivative of the function we are trying to find a root for. So $f'(x)$ will not be well behaved when $g'(x)$ too flat, exactly the problem we observed when $\mu_0 > 15.01026$ in the example.

However, around the root, $g'(x)$ is bounded away from zero, so the procedure should work well.

The bottom line is that often you need to be careful about where you start Newton-Raphson and also you need to monitor how it is converging (to be addressed later).

Convergence rates

Also of interest, is how fast a root finding scheme converges to a root.

As we've seen so far, the bisection method and functional iteration both have linear convergence.

Procedures that have linear convergence satisfy

$$\lim_{n\to\infty} \frac{e_n}{e_{n-1}} = \lambda$$

where $e_n = x_n - x_\infty$. Assuming that the procedure can be written in the form

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

we can look at a Taylor series approximation

$$\begin{aligned} e_n &= f(x_{n-1}) - f(x_\infty) \\ &= f'(z)e_{n-1} \end{aligned}$$

where z between x_{n-1} and x_∞. Provided that $f'(z)$ is continuous and x_0 isn't too far from x_∞, this implies that

$$\lim_{n\to\infty} \frac{e_n}{e_{n-1}} = f'(x_\infty)$$

As we saw last time, if $f'(x_\infty)$ is bounded between -1 and 1, this implies that the scheme will converge to a fixed point.

However for Newton-Raphson

$$f'(x_\infty) = \frac{g(x_\infty)g''(x_\infty)}{g'(x_\infty)^2} = 0$$

which suggests that it should converge at a faster rate. Note that we have to be a bit careful here, as we can get into division by 0 issues due to $g'(x)$.

Let $e_n = x_n - x_\infty$ be the current approximation error. Then a Taylor series approximation gives

$$\begin{aligned} e_n &= f(x_{n-1}) - f(x_\infty) \\ &= f'(x_\infty)e_{n-1} + \frac{1}{2}f''(z)e_{n-1}^2 \\ &= \frac{1}{2}f''(z)e_{n-1}^2 \end{aligned}$$

where z between x_{n-1} and x_∞. Provided that $f''(z)$ is continuous and x_0 isn't too far from x_∞, this implies that Newton converges. In addition, this implies that

$$\lim_{n\to\infty} \frac{e_n}{e_{n-1}^2} = \frac{1}{2} f''(x_\infty)$$

Newton-Raphson has what is known as quadratic convergence. In general, a scheme converges at order α if

$$\lim_{n\to\infty} \frac{e_n}{e_{n-1}^\alpha} = \lambda \neq 0$$

Note that α does not need to be an integer. For example, the Illinois scheme converges with order 1.442 (Thisted, 1988). The secant method, which is to come, converges at a rate between 1 and 2.

Assessing convergence

While with the bisection method, you can pre-specify the number of iterations needed to reach a desired level of accuracy, other algorithms such as Newton-Raphson, you can't.

Instead the sequence of $|x_n - x_{n-1}|$ is monitored. When $|x_n - x_{n-1}|$ gets small enough (say < TOL), the procedure is stopped.

The choice of TOL depends the level of accuracy desired and the magnitude of x_∞.

For example setting TOL = 0.1 when $x_\infty = 0.001$ is a bit useless. As an alternative, a stopping criteria of the form

$$\frac{|x_n - x_{n-1}|}{|x_n|} < \text{TOL}$$

is sometimes used. You need to be a bit careful when x_n is around 0 with this relative error criterion.

One other issue when using stopping criteria like either of the above is when the procedure converges very slowly or diverges. Usually a maximum number of iterations needs to be specified.

The following segment of Matlab code gives the basic form for most iterative root finding routines.

```
% Initilize loop variables

i = 0;
diff = 2 * TOL;  % guarantees at least one pass
                 % through loop

while (i < Nmax && diff > TOL)  % Not converged
  i = i + 1;
  xold = xnew;
  xnew = f(xold);
  diff = abs(xold - xnew);
end

if (diff > TOL)
  warning('Method did not converage after Nmax
steps');
end
```

In addition, it also useful to examine $g(x_{last})$, the value of the function at the output of the root finding routine to make sure that you are close enough to the root. You could have a function such at $g(x_{n-1})$ is a bit away from zero, but $-g(x_{n-1})/g'(x_{n-1}) = x_n - x_{n-1}$ is close to zero.

Advantages of Newton-Raphson

- Fast – quadratic convergence.
- When used to optimize a function, can also get variance of estimate.

$$l(\mu) = \frac{d \log L(\mu)}{d\mu}$$

$$l'(\mu) = \frac{d^2 \log L(\mu)}{d\mu^2}$$

The observed information is given by $-l'(\mu_\infty)$ and its inverse is a common variance estimate for μ_∞.

- Easily extended to multi-parameter problems.

Disadvantages/Problems with Newton-Raphson

- Doesn't have to converge. However modifications can be made to avoid non-convergence problems (e.g. take smaller steps).

- Uses derivatives, which can have a high computational burden. However, in cases where derivatives may be difficult to deal with, the derivatives can be numerically approximated.

Secant Method

An approach which uses a numerical approximation to the derivative as part of the routine. Uses the idea

$$g'(x_{n-1}) \approx \frac{g(x_{n-2}) - g\left(x_{n-1}\right)}{x_{n-2} - x_{n-1}}$$

if x_{n-2} and x_{n-1} aren't too far apart. This leads to the updating formula of

$$x_n = x_{n-1} - \frac{g(x_{n-1})(x_{n-1} - x_{n-2})}{g(x_{n-1}) - g\left(x_{n-2}\right)}$$

Now the secant method has similar properties to Newton-Raphson. One difference is that it doesn't have quadratic convergence, but it is still better than linear. It can be shown that

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n e_{n-1}} = \frac{g''(x_\infty)}{2g'(x_\infty)}$$

You do need to have a reasonable convergence criterion for this procedure as if you take the algorithm too far, $x_{n\text{-}2} = x_{n\text{-}1}$ (and $g(x_{n-1}) = g(x_{n-2})$), so eventually you will get a division by 0 problem.

For the variance heterogeneity example the two procedures converge similarly. Setting $\mu_0 = \bar{y}$ for both procedures (and $\mu_{-1} = \bar{y} - 0.1$ for the secant procedure) and TOL = 10^{-6}), they both converge to the same point $\hat{\mu}$ = 9.8279 with I_{obs} = 0.7627, which gives Var($\hat{\mu}$) = 1.3112.

In addition,

$$\hat{\mu}_{NR} - \hat{\mu}_{SEC} = \text{-1.1191e-013}$$

$$I_{NR} - I_{SEC} = \text{5.2013e-006}$$

Newton-Raphson took 6 iterates to converge and Secant took 8 iterates.

The path to convergence is not the same for the two algorithms

Iteration	Newton	Secant
0	11.7646	11.7646
1	8.4280	8.5143
2	9.4024	10.4980
3	9.7830	10.0452
4	9.8274	9.7867
5	9.8279	9.8303
6	9.8279	9.8279

To see the advantage of quadratic convergence, it would take the Bisection algorithm around 22 iterations to reach the same accuracy (with $b_0 - a_0 = 6$).