Newton – Raphson

Probably the most popular root finding method.

Based on Taylor series approximation

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

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

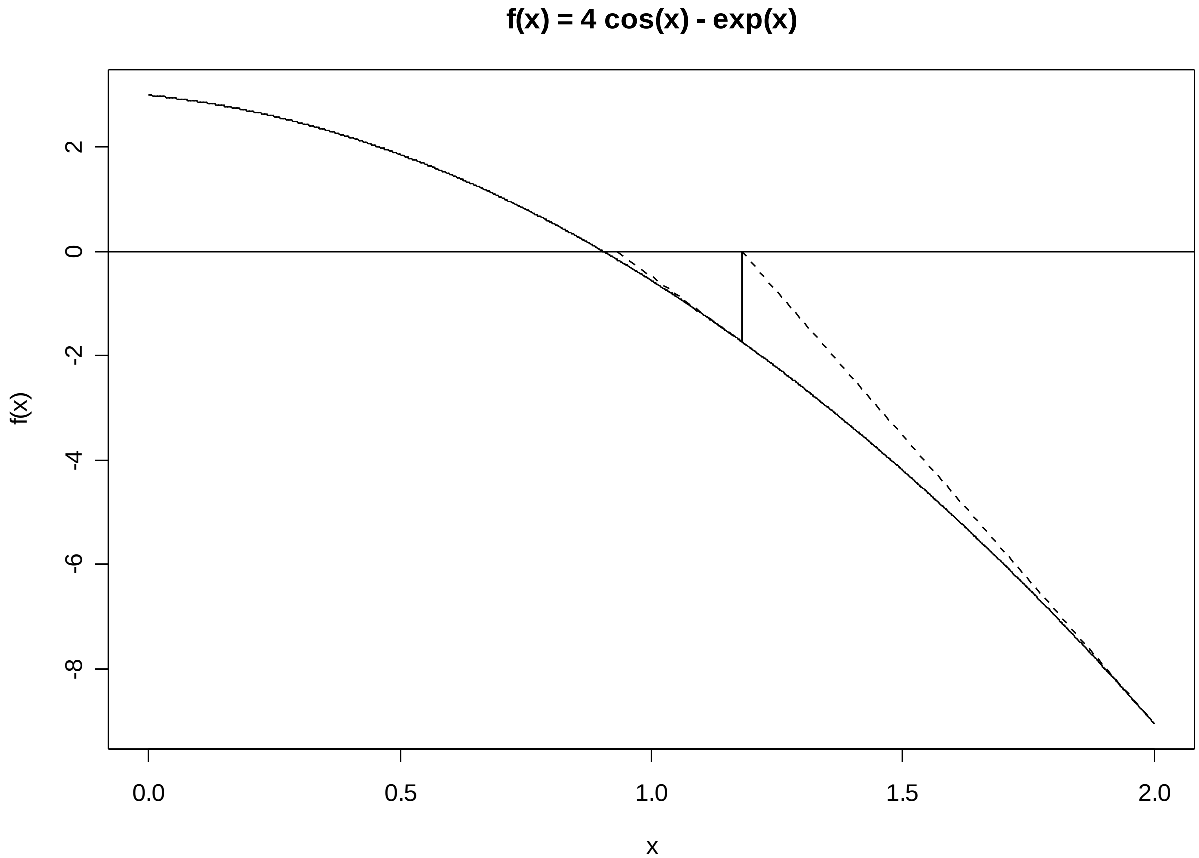
$$x_n = x_{n-1} - \frac{g(x_{n-1})}{g'(x_{n-1})}$$

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

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

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

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

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

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

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

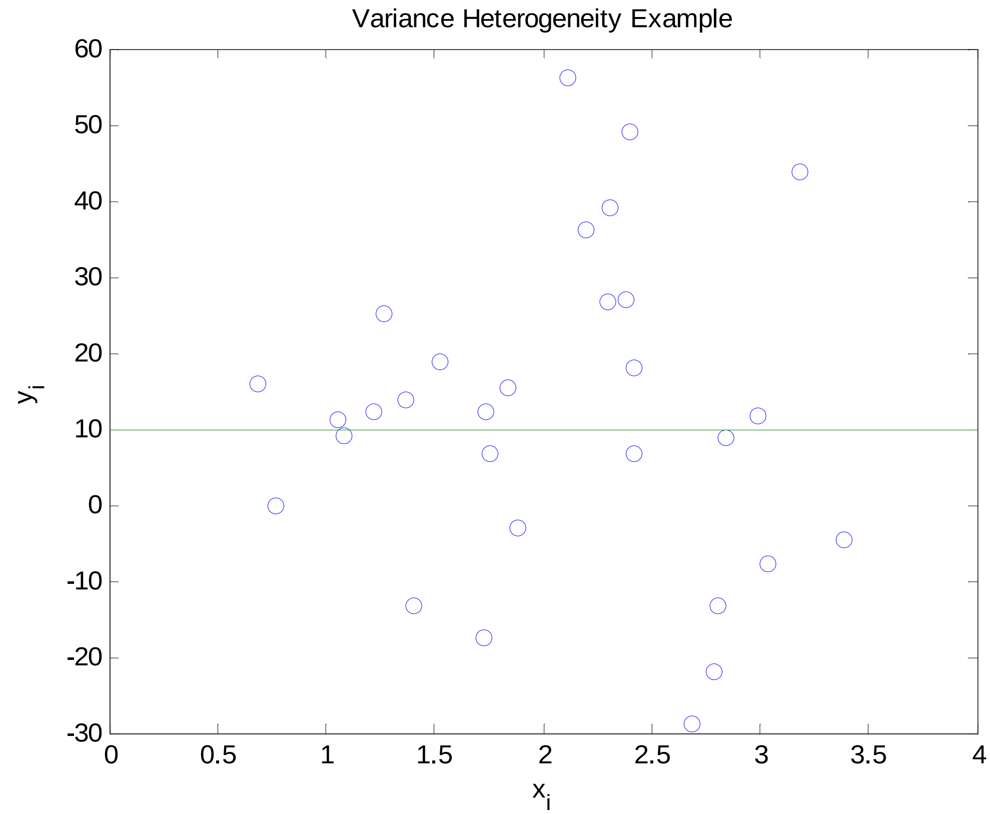
$$= \frac{n}{\mu^2} - \frac{3}{\mu^4} \sum_{i=1}^{n} \left(\frac{y_i - \mu}{x_i} \right) \frac{y_i}{x_i} - \frac{1}{\mu^3} \sum_{i=1}^{n} \frac{y_i}{x_i^2}$$

The Newton scheme for this problem is given by

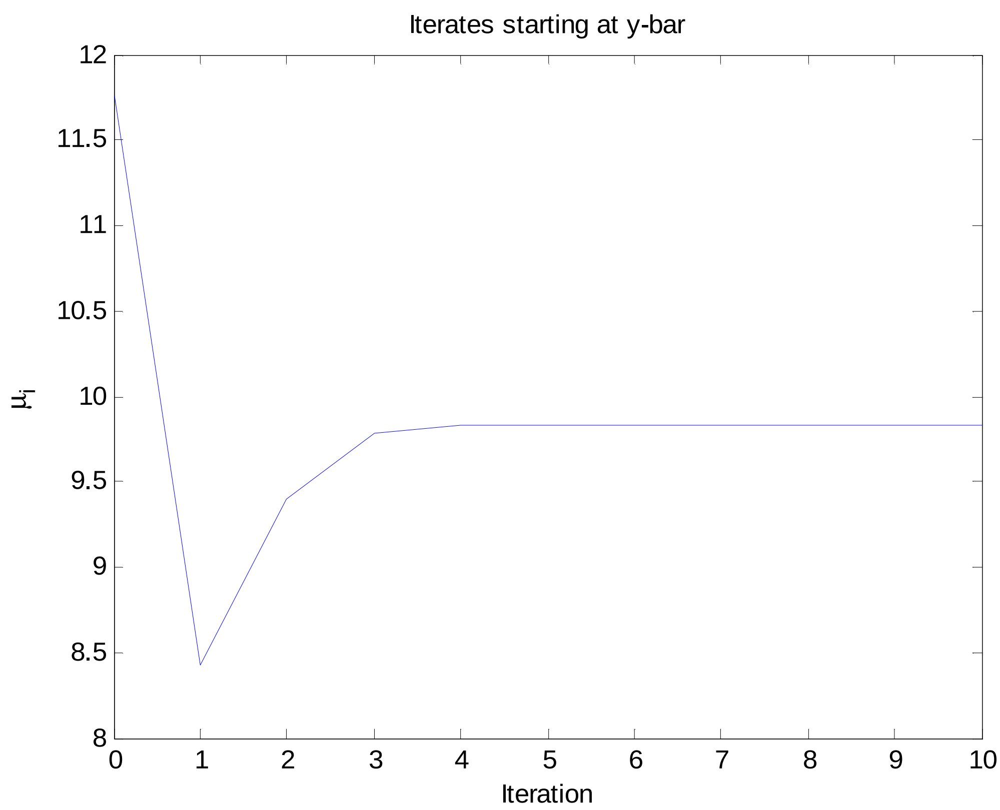
$$\mu_n = \mu_{n-1} - \frac{l(\mu_{n-1})}{l'(\mu_{n-1})}$$

As an example, 30 independent observations were generated in Matlab from $Y_i \left| x_i \sim N(10, 10^2 x_i^2) \right.$ 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.

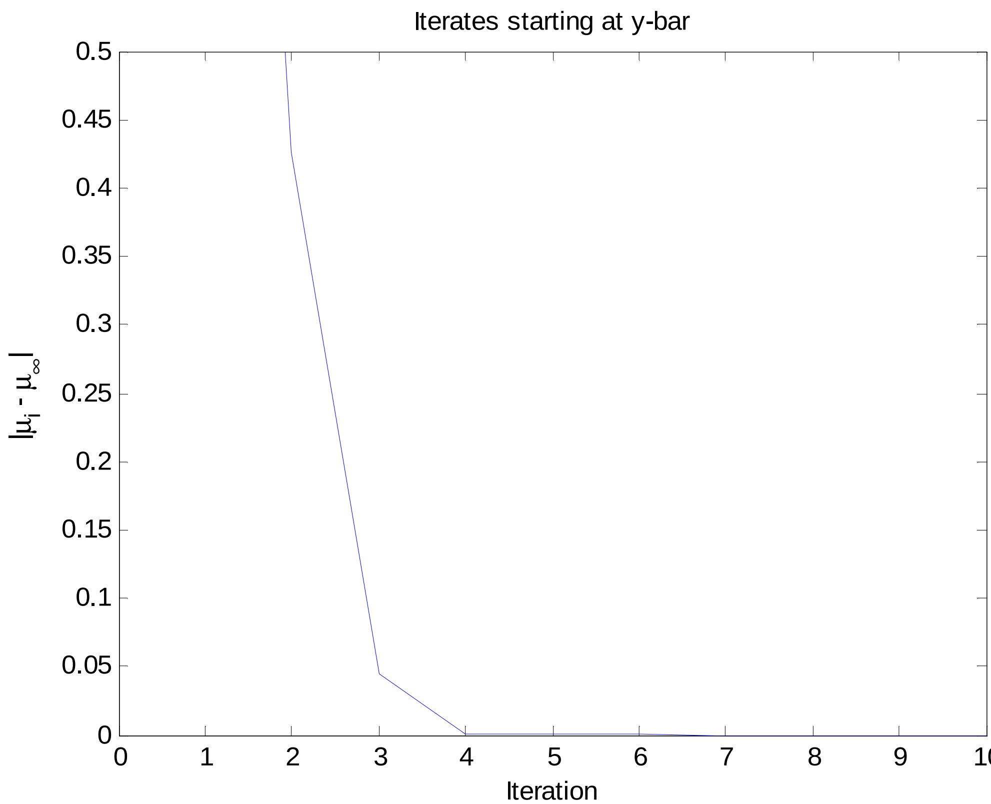


Variance Heterogeneity Example
60
50
40
30
20
10
0
-10
-20
-30
0
0.5
1
1.5
2
2.5
3
3.5
4
y_i
x_i



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 $\mu_0 = 15.1$. The sequence of iterates quickly diverges.

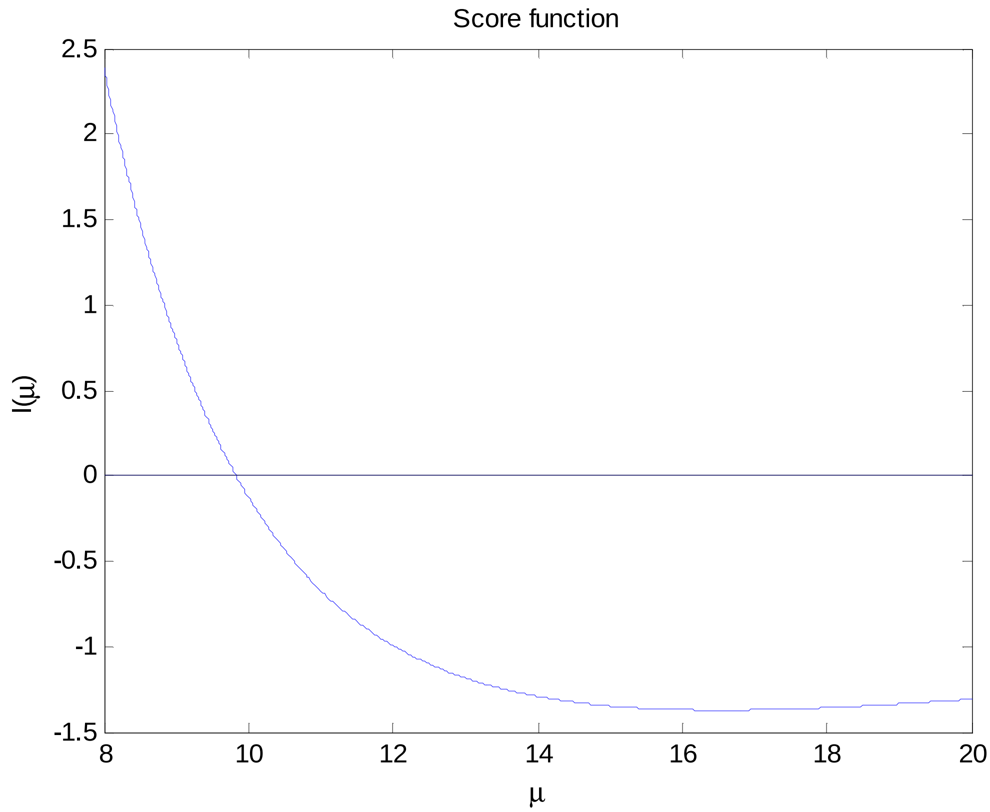
Iteration	μ_i
0	15.1000
1	-24.8822
2	335.4398
3	667.3233

However if we start close by at $\mu_0 = 15$, 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

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

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

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

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 to far, $x_{n-2} = x_{n-1}$ (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,

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

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

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 $b_0 - a_0 = 6$).