CHAPTER SEVENTEEN

Solving Nonlinear Algebraic Equations

Synopsis

"A view of nature as dense and nonlinear is at the core of our contemporary science. Process and order emerge subtly." — Gregory Benford

In this chapter, we describe leading methods for solving algebraic nonlinear equations.

ROOT FINDERS

ROOT FINDING OF a nonlinear equation is an important exercise in science and engineering. Some applications requiring roots are as follows: computation of roots of polynomials for Gauss quadrature; finding fixed points of nonlinear equations and nonlinear maps; finding minima or maxima during optimisation.

In this chapter, we will discuss how to find roots for a single nonlinear equation. The methods discussed here can be generalised to multiple nonlinear equations. In Section Solution of Algebraic Equations, we will discuss methods for solving a set of linear equations.

In this chapter, we will describe the following root-finding algorithms: *Bisection method*, *Newton-Raphson's method*, *Secant method*, and *Relaxation method*. Our objective is to solve equations of the form $f(x) = 0$. We start with the bisection method.

Bisection Method

Consider one of the roots of equation $f(x) = 0$. In bisection method, we adopt the following strategy:

1. We plot $f(x)$ vs. x and identify two points x_l and x_r that are on the left and right sides of the root x^*. Thus, we bracket the root between x_l and x_r. See Figure 108 for an illustration.
2. We compute the mid point $x = (x_l+x_r)/2$.
3. If $|f(x)| \le \varepsilon$, where ε is the tolerance level, then we have found the root. We stop here.
4. Otherwise, $|f(x)| > \varepsilon$. Now, if $x < x^*$, x replaces x_l, otherwise x replaces x_r.
5. We repeat the above process till $|f(x)| \le \varepsilon$.

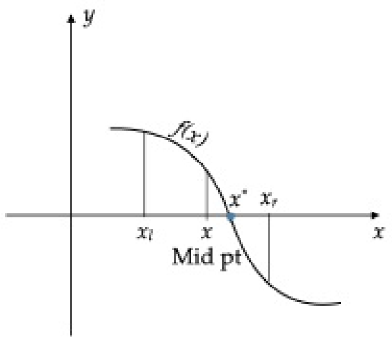

Figure 108: Root finder using bisection method. We start with x_l and x_r, and approach the root iteratively.

Now we estimate the error in the bisection method. Let us denote the two starting points as x_0 and x_1, and later x's *as* $x_2, x_3, \ldots x_n$, etc. Note that

$x_{n+1} = (½)\ (x_n+x_{n-1})$. …..(100)

Hence, the error, $\varepsilon_n = x\backslash * - x_n$, evolves as

$\varepsilon_{n+1} = (½)\ (\varepsilon_n+\varepsilon_{n-1})$.

Hence, the error, which is average of the two previous steps, converges slowly.

Example 1: Using the bisection method, we find root of the equation $f(x)$ = exp(x)−6. The correct answer is log(6) ≈ 1.791759469228055. First, we plot the function $f(x)$ in Figure 109(a), and choose $x_0 = 3$ and $x_1 = 1$, which are on the opposite sides of the root. After this, we compute x_n's iteratively using Eq. (100). We plot x_n vs. n in Figure 109(b), according to which, for a tolerance level of ε = 0.001, the iteration converges to 1.7890625 in 9 steps. A code segment is given below:

```
def f(x):
    return np.exp(x)-6

eps = 0.001; N = 10
x = np.zeros(N); x[0] = 3; x[1] = 1

for i in range(2,N):
    mid = (x[i-2] + x[i-1])/2
    if (abs(mid-x[i-1]) < eps):
        x[i] = mid
        break
    if f(x[i-2])*f(mid) > 0:
        x[i] = mid
    else:
        x[i-1] = x[i-2]; x[i] = mid
    print("i, mid = ", i, mid)

print ("estimated root ", mid)
```

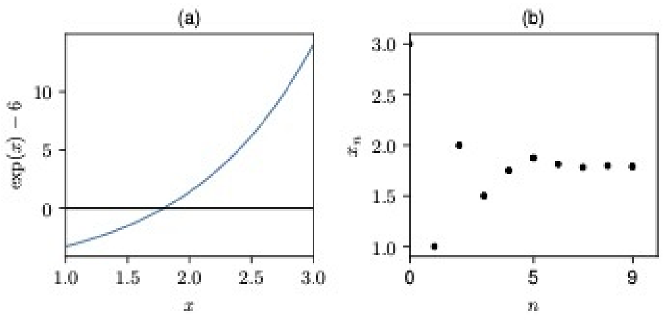

Figure 109: Example 1: Finding the root of $f(x)$ = exp(x)−6 using the bisection method. (a) plot of $f(x)$ vs. x. (b) Plot of x_n vs. n.

Example 2: We employ bisection method to compute the root of $f(x) = x^{1/3}$. We choose $x_0 = 1.7$ and $x_1 = -0.4$ and iterate Eq. (100). For a precision of 10^{-5}, the solution converges to x* ≈ 0 in 18 iterations.

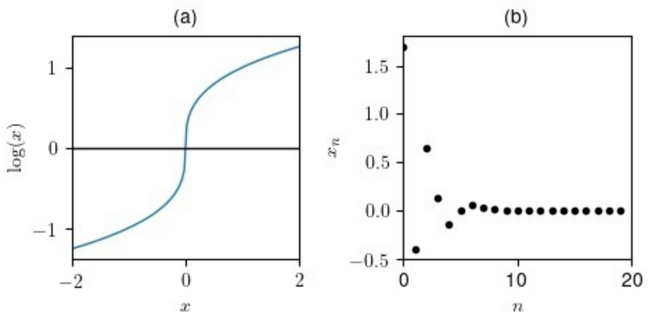

Figure 110: Example 2: Finding the root of $f(x) = x^{1/3}$ using bisection method. (a) plot of $f(x)$ vs. x. (b) Plot of x_n vs. n.

Newton-Raphson Method

The Newton-Raphson method provides a faster convergence to the root than the bisection method. In this method, we start with x_0 and compute x_1 using the slope at x_0. As shown in Figure 111,

$$f'(x_0) = \frac{f(x_0)}{x_0 - x_1}, \text{ hence, } x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

For the (n+1)th step,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad \ldots(101)$$

The process is continued until $|x_{n+1} - x_n| < \varepsilon$, or till the tolerance level is met.

Now let us analyse the error for this method. At the root x*, $f(x^*) = 0$. Rewriting of Eq. (101) and its Taylor expansion around x* yields

$$x^* - x_{n+1} = x^* - x_n + \frac{f(x^*) - \epsilon_n f'(x^*) + (1/2)\epsilon_n^2 f''(x^*) + H.O.T.}{f'(x^*) - \epsilon_n f''(x^*) + H.O.T.},$$

$$\text{or, } \epsilon_{n+1} \approx -\frac{1}{2}\epsilon_n^2 \frac{f''(x^*)}{f'(x^*)}, \quad \ldots..(102)$$

where *H.O.T.* stands for higher order terms. In the above analysis, the denominator too is expanded using Taylor series. Thus, the error converges as ε_n^2 as long as $f''(x^*)/f'(x^*)$ is finite.

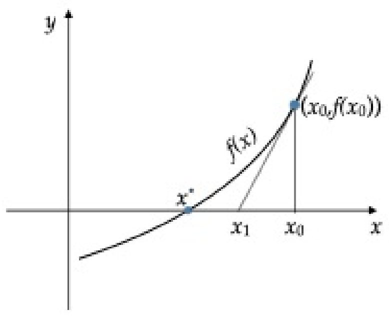

Figure 111: Root finder using Newton-Raphson method.

The quadratic convergence of Newton-Raphson method is a faster than the linear convergence of bisection method. However, Newton-Raphson method fails to converge if $|f''(x^*)/f'(x^*)|$ is large. For this case, the iteration diverges, that is, $\varepsilon_{n+1} > \varepsilon_n$. See Figure 112 for an illustration.

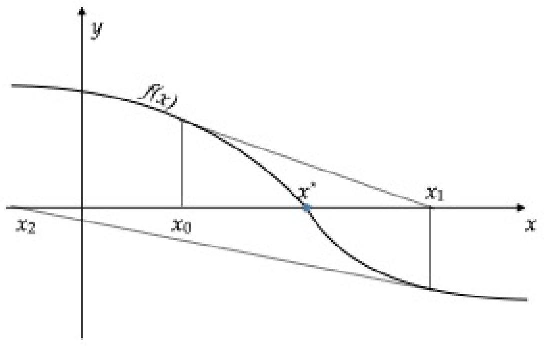

Figure 112: Newton-Raphson method fails to converge when $|f''(x^*)/f'(x^*)|$ is large. In this case, we go further away from the root, or $\varepsilon_{n+1} > \varepsilon_n$.

Example 3: We solve for the root of $f(x) = \exp(x) - 6$ using Newton-Raphson method. We start with $x_0 = 3$ and iterate using Eq. (101). For the tolerance level of $\varepsilon = 0.001$, the iteration converges to 1.7917594693651107 in 5 iteration. See Figure 113(a) for an illustration. A code segment for the computation is given below:

```
for i in range(1,N):
….x[I] = x[i-1] - f(x[i-1])/np.exp(x[i-1])
….if (abs(x[i]-x[i-1]) < eps):
```

```
…….break
print ("estimated root ", x[i])
```

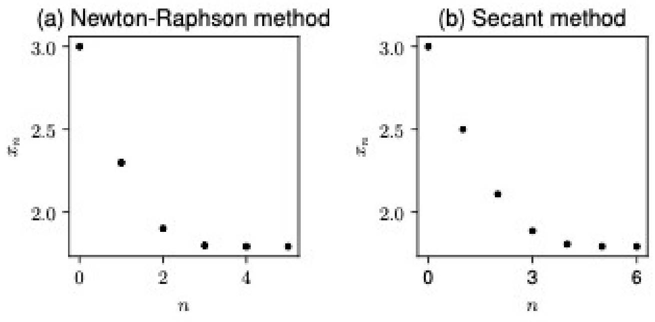

Figure 113: Finding the root of $f(x) = \exp(x) - 6$. Plots of x_n vs. n for (a) Newton-Raphson method (Example 3); (b) secant method (Example 5).

Example 4: Newton-Raphson method fails to compute the root of $f(x) = x^{1/3}$. We start with $x_0 = 1$ and iterate Eq. (101). As shown in Figure 114(a), the solution x_n diverges from the root $x^* \approx 0$. This is because of the divergence of $|f''(x^*)/f'(x^*)|$ near the origin.

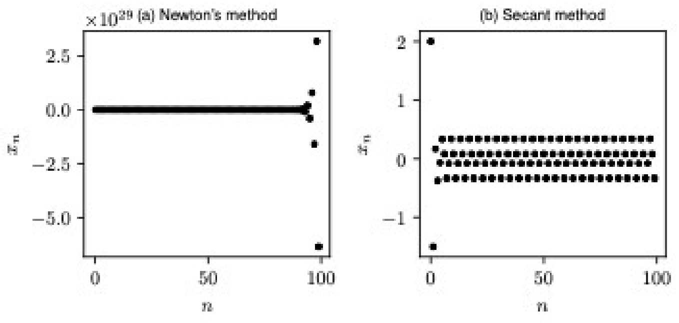

Figure 114: For $f(x) = x^{1/3}$. Plots of x_n vs. n for (a) Newton-Raphson method (Example 4); (b) secant method (Example 6). Both these methods fail to find the roots of the equation.

Secant Method

Often, it may be impractical to compute the derivative of a function. It is specially so when *f*(*x*) is provided as a time series. In that case, we estimate *f* '(x_n) of Eq. (101) using two neighbours (x_{n-1}, $f(x_{n-1})$) and (x_n, $f(x_n)$) (see Figure 115):

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

Substitution of the above *f* '(x_n) in Eq. (101) yields

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} . \quad .....(103)$$

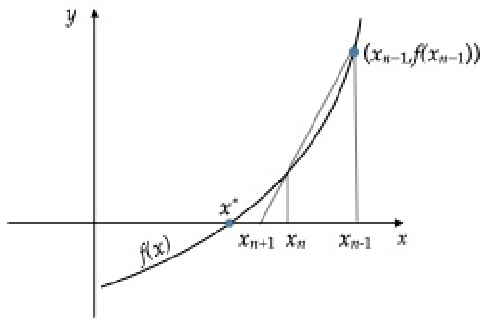

Figure 115: Root finder using Secant method.

This method is called secant method due to the secant passing through the points (x_n,$f(x_n)$) and (x_{n-1},$f(x_{n-1})$). Also, the convergence of the secant method is similar to that of Newton-Raphson method, that is, errors converge as in Eq. (102).

Example 5: We solve for the root of $f(x) = \exp(x) - 6$ using the secant method. We start with $x_0 = 3$ and $x_1 = 2.5$, and take tolerance level of $\varepsilon = 0.001$. The iteration of Eq. (103) converges to 1.791764077555014 in 6 steps. See Figure 113(b) for an illustration.

Example 6: The secant method fails to compute the root of $f(x) = x^{1/3}$ because $|f''(x^*)/f'(x^*)|$ diverges near the origin. However, unlike Newton-Raphson method, x_n's oscillate around 4 values near the origin. See Figure 114(b) for an illustration.

Relaxation Method

Often we need to solve equations of the form $f(x) = x$, which is equivalent to finding the roots of $F(x) = f(x) - x$. Let us denote the solution of $f(x) = x$ by x*, that is, $f(x\backslash) = x\backslash$. Following the notation of dynamical systems, we refer to $x\backslash*$ as a *fixed point* (*FP*).

To find $x*$, we start with $x = x_0$ and compute $x_1 = f(x_0)$. After this we compute $x_2 = f(x_1)$, and continue further. The $(n+1)$th step is

$$x_{n+1} = f(x_n) \text{ . .....(104)}$$

For some fixed points, called *stable fixed points*, the iteration converges, that is,$|x_{n+1} - x_n| < \varepsilon$. The fixed points for which the iteration does not converge are called *unstable fixed points*. Figure 115 illustrates these points. The left FP is stable, while the one in the middle is unstable. The diagram of Figure 115 is also called a *web diagram* because the arrows form a web.

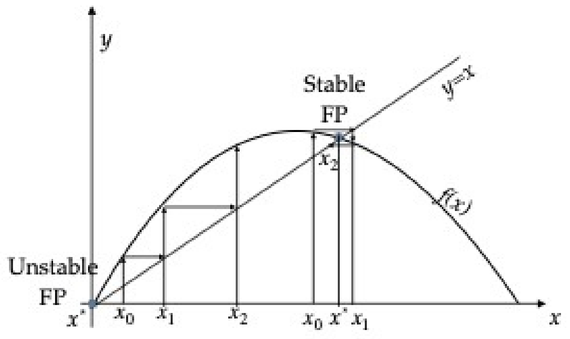

Figure 116: The web diagram for finding the fixed points of $f(x) = x$ using the relaxation method.

The relaxation method has a simple criteria for convergence. We show below that the iterations converge to the *stable fixed point* $x\backslash*$ when $|f'(x\backslash)| < 1$. *On the other hand, the iterations diverge when* $|f'(x^{**})| > 1$. We prove the above statement using the following arguments.

We subtract $x*$ from Eq. (104) that yields

$x_{n+1} - x^* = f(x_n) - x^* = f(x_n) - f(x^*)$.

Using Taylor expansion of $f(x_n)$ around x^* and ignoring the higher order terms, we obtain

$x_{n+1} - x^* = (x_n - x^*)f'(x\backslash *)$,

or, $\varepsilon_{n+1} = \varepsilon_n f'(x\backslash *)$.

Clearly, $|\varepsilon_{n+1}| < |\varepsilon_n|$ when $|f'(*x*)| < 1$*. Hence,* |f'*(*x**)*| < 1 *is the necessary condition for the convergence of the relaxation method. The iteration process diverges when* |f'*(*x**)| > 1.

How do we compute the unstable FP considering that the iterations diverge for this case? Figure 116 provides us a hint. The reversed arrows converge to the unstable fixed point. Hence, the iteration

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

will lead us to the unstable FP. Note that reversal of the arrows is equivalent to going back in time.

Example 7: We find the solutions of $f(x) = 2.5\, x(1-x) = x$ (see Figure 117(a)) using the relaxation method. This equation has two solutions: $x^* = 0$ and $x^* = 1-1/2.5 = 0.6$. Note that $f'(0) = 2.5$ and $f'(0.6) = -0.5$. Hence, $x^* = 0$ is an unstable FP, while $x^* = 0.6$ is a stable FP. To reach to the stable FP, we start with $x_0 = 0.1$ and iterate that takes us to $x^* = 0.6$. The small black dots of Figure 117(b) represent x_n's during the iteration.

To obtain the unstable FP using relaxation method, we start with $x_{n+1} = 0.2$ and iterate backwards, that is, $x_n = f^{-1}(x_{n+1})$. We finally reach near the FP, $x^* = 0$. The large red circles of Figure 117(b) represent x_n for this iteration. Note that $f^{-1}(y)$ has two solutions. However, we employ

$$x = f^{-1}(y) = \frac{a - \sqrt{a^2 - 4ay}}{2a}$$

with *a* = 2.5; this solution takes us to *x** = 0.

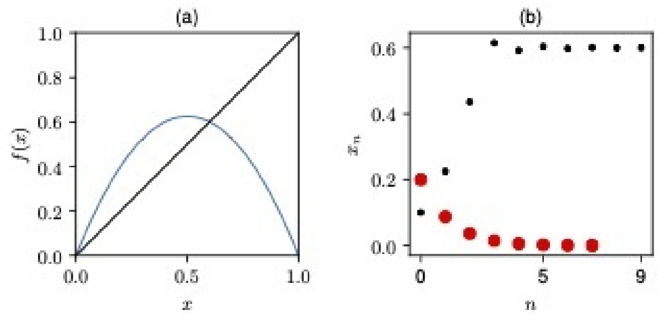

Figure 117: Example 7: Finding the roots of *f*(*x*) = 2.5 *x*(1−*x*). (a) Plot of *f*(*x*) vs. *x*. (b) Plot of x_n vs. *n*; small black dots converge to the root *x** = 0.6, while the large red circles converge to *x** = 0.

Python’s root finder

Python’s scypy module has several functions for finding the roots of equations. These functions are *brentq*, *brenth*, *ridder*, and *bisect*, whose arguments are the function and the bracketing interval. Here we discuss how to use the function *brentq*. The following code segment finds the roots of $g(x) = x^2 - 1$ and $f(x) = \sin(\pi x) - x$.

```
In [15]: from scipy import optimize

In [308]: g = lambda x: x**2 - 1

In [309]: optimize.brentq(g, 0.5, 1.5)       # finds the root in
the interval [0.5, 1.5]
Out[309]: 0.9999999999999993

In [310]: optimize.brentq(g, -1.5, -0.5)
Out[310]: -0.9999999999999993

In [312]: f =lambda x: np.sin(np.pi*x) - x

In [313]: optimize.brentq(f,-0.1, 0.1)
Out[313]: 0.0

In [314]: optimize.brentq(f,0.5, 1)
Out[314]: 0.7364844482410411
```

Assessment of Root Finders

Let us review the root finders discussed in this chapter. The bisection method is quite robust and simple, and it is guaranteed to converge. The convergence however is slow. On the other hand, Newton-Raphson and secant methods converge quickly. These two methods however fail if $|f''(x^*)/f'(x^*)|$ diverges. We illustrate this issue using $f(x) = x^{1/3}$ as an example. The bisection method can find the root of $f(x) = x^{1/3}$, but Newton-Raphson and secant methods fail to do so.

The relaxation method is an efficient and direct method to find the solution of $f(x) = x$. This method is very useful in dynamical systems.

We will employ root finders in the next chapter where we solve boundary value problems.

****

Conceptual Questions

1. Provide examples in physics where we need to find roots of nonlinear equations.
2. What are the merits and demerits of the bisection, Newton-Raphson, and secant methods for finding roots of nonlinear equations.

Exercises

1. Find root of $f(x) = \log(x)$ using the bisection, Newton-Raphson, and secant methods. Compare their convergence rates.
2. Find all the roots of $f(x) = 65\,x^5 - 70\,x^3 + 15x$ numerically.
3. Give an example of a function for which Newton-Raphson method fails to find roots.
4. Find the solutions of $\sin(\pi x) = x$ using relaxation method.
5. Find the root of $f(x) = \tan(2x) - x$ in the interval $[-1/2, 1]$ using relaxation and Newton-Raphson methods.
6.