

MA3105

Numerical Analysis

Autumn 2021

Satvik Saha
19MS154

*Indian Institute of Science Education and Research, Kolkata,
Mohanpur, West Bengal, 741246, India.*

Contents

1	Time complexity	1
1.1	Runtime cost	1
1.2	Asymptotic growth	2
2	Root finding methods	4
2.1	Tabulation method	4
2.2	Bisection method	5
2.3	Newton-Raphson method	5
2.4	Secant method	6
2.5	Fixed point method	6

1 Time complexity

1.1 Runtime cost

When designing or implementing an algorithm, we care about its efficiency – both in terms of execution time, and the use of resources. This gives us a rough way of comparing two algorithms. However, such metrics are architecture and language dependent; different machines, or the same program implemented in different programming languages, may consume different amounts of time or resources while executing the same algorithm. Thus, we seek a way of measuring the ‘cost’ in time for a given algorithm.

For example, we may look at each statement in a program, and associate a cost c_i with each of them. Consider the following statements.

```
one = 1;           // c_1
two = 2;           // c_2
three = 3;         // c_3
```

The total cost of running these statements can be calculated as $T = c_1 + c_2 + c_3$, simply by adding up the cost of each statement. Similarly, consider the following loop construct.

```

sum = 0;                                // c_1
for (i = 0; i < n; i++)                  // c_2
    sum += a[i];                          // c_3

```

The total cost can be shown to be $T(n) = c_1 + c_2(n+1) + c_3n$; this time, we must take into account the number of times a given statement is executed. Note that this is linear. Another example is as follows.

```

sum = 0;                                // c_1
for (i = 0; i < n; i++)                  // c_2
    for (j = 0; j < n; j++)              // c_2
        sum += a[i][j];                  // c_4

```

The total cost can be shown to be $T(n) = c_1 + c_2(n+1) + c_3n(n+1) + c_4n^2$. Note that this is quadratic. Finally, consider the following recursive call.

```

int factorial (int n) {                  // c_1
    if (n == 0)                          // c_2
        return 1;                        // c_3
    return n * factorial(n - 1);          // c_4
}

f = factorial(n);                        // c_5

```

The cost can be shown to be $T(n) = c_5 + (c_1 + c_2)(n+1) + c_3 + c_4n$. This turns out to be linear.

In all these cases, we care about our total cost as a function of the input size n . Moreover, we are interested mostly in the *growth* of our total cost; as our input size grows, the total cost can often be compared with some simple function of n . Thus, we can classify our cost functions in terms of their asymptotic growths.

1.2 Asymptotic growth

Definition 1.1. The set $O(g(n))$ denotes the class of functions f which are asymptotically bounded above by g . In other words, $f(n) \in O(g(n))$ if there exists $M > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$|f(n)| \leq Mg(n).$$

This amounts to writing

$$\limsup_{n \rightarrow \infty} \frac{|f(n)|}{g(n)} < \infty.$$

Example. Consider a function defined by $f(n) = an + b$, where $a > 0$. Then, we can write $f(n) \in O(n)$. To see why, note that for all $n \geq 1$, we have

$$|f(n)| = |an + b| \leq an + |b| \leq (a + |b|)n.$$

Thus, setting $M = a + |b| > 0$ completes the proof.

Example. Consider a polynomial function defined by

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0,$$

with some non-zero coefficient. Then, we can write $f(n) \in O(n^k)$. Like before, note that for all $n \geq 1$, we have

$$|f(n)| \leq \sum_{i=0}^k |a_i| n^i \leq \sum_{i=0}^k |a_i| n^k = (|a_k| + |a_{k-1}| + \cdots + |a_0|) n^k.$$

Thus, setting $M = |a_k| + \cdots + |a_0| > 0$ completes the proof.

Theorem 1.1. If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then

$$f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\}).$$

Definition 1.2. The set $\Omega(g(n))$ denotes the class of functions f are asymptotically bounded below by g . In other words, $f(n) \in \Omega(g(n))$ if there exists $M > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$|f(n)| \geq M g(n).$$

This amounts to writing

$$\liminf_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0.$$

Definition 1.3. The set $\Theta(g(n))$ denotes the class of functions f which are asymptotically bounded both above and below by g . In other words, $f(n) \in \Theta(g(n))$ if there exist $M_1, M_2 > 0$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$M_1 g(n) \leq |f(n)| \leq M_2 g(n).$$

This amounts to writing $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Another class of notation uses the idea of dominated growth.

Definition 1.4. The set $o(g(n))$ denotes the class of functions f which are asymptotically dominated by g . In other words, $f(n) \in o(g(n))$ if for all $M > 0$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$|f(n)| < M g(n).$$

This amounts to writing

$$\lim_{n \rightarrow \infty} \frac{|f(n)|}{g(n)} = 0.$$

Definition 1.5. The set $\omega(g(n))$ denotes the class of functions f which asymptotically dominate g . In other words, $f(n) \in \omega(g(n))$ if for all $M > 0$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$|f(n)| > Mg(n).$$

This amounts to writing

$$\lim_{n \rightarrow \infty} \frac{|f(n)|}{g(n)} = \infty.$$

Definition 1.6. We say that $f(n) \sim g(n)$ if f is asymptotically equal to g . In other words, $f(n) \sim g(n)$ if for all $\epsilon > 0$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$\left| \frac{f(n)}{g(n)} - 1 \right| < \epsilon.$$

This amounts to writing

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1.$$

We often abuse notation and treat the following as equivalent.

$$T(n) \in O(g(n)), \quad T(n) = O(g(n)).$$

2 Root finding methods

Consider an equation of the form $f(x) = 0$, where $f: [a, b] \rightarrow \mathbb{R}$ is given. We wish to solve this equation, i.e. find the roots of f .

Note that for *arbitrary* functions, this task is impossible. To see this, consider a function f which assumes the value 1 on $[0, 1] \setminus \{\alpha\}$ and $f(\alpha) = 0$, for some $\alpha \in [0, 1]$. There is no way of pinpointing α without checking f at every point in $[0, 1]$. Besides, a computer cannot reasonably store real numbers with arbitrary precision.

Thus, we direct our attention towards *continuous* functions f . We only seek sufficiently accurate approximations of its root $\alpha \in (a, b)$.

Theorem 2.1 (Intermediate Value Theorem). *Let $f: [a, b] \rightarrow \mathbb{R}$ be continuous. If $f(a)f(b) < 0$, then there exists $\alpha \in (a, b)$ such that $f(\alpha) = 0$.*

2.1 Tabulation method

To identify the location of a root of f on an interval $I = [a, b]$, we subdivide I into n subintervals $[x_i, x_{i+1}]$ where $x_i = a + (b - a)i/n$. Now, we simply apply the Intermediate Value Theorem to f on each of these intervals. If $f(x_i)f(x_{i+1}) < 0$, then f has a root somewhere in (x_i, x_{i+1}) . Note that the error in our approximation is on the order of $|b - a|/n$. The precision of this method can be improved by increasing n .

To reach a degree of approximation ϵ , we must iterate n times, where

$$n > \frac{b - a}{\epsilon}.$$

2.2 Bisection method

Here, we first verify that $f(a)f(b) < 0$, thus ensuring that f has a root within (a, b) . Now, set $x_1 = a + (b - a)/2$ and apply the Intermediate Value Theorem on the subintervals $[a, x_1]$ and $[x_1, b]$. One of these *must* contain a root of f . Note that if $f(x_1) = 0$, we are done; otherwise, let $I_1 = [a_1, b_1]$ be the subinterval containing the root. Repeat the above process, obtaining successive subintervals I_n with lengths $|b - a|/2^n$. The error in our approximation is of this order, and can be controlled by stopping at appropriately large n .

The quantity $x_{n+1} = (a_n + b_n)/2$ is a good approximation for the actual root α since we know that $x_{n+1}, \alpha \in [a_n, b_n]$, so

$$|x_{n+1} - \alpha| \leq |b_n - a_n| = 2^{-n}|b - a| \rightarrow 0.$$

To reach a degree of approximation ϵ , we must iterate n times, where

$$n > \log_2 \frac{b - a}{\epsilon}.$$

2.3 Newton-Raphson method

Assuming that f is twice differentiable, use Taylor's theorem to write

$$f(x) = f(x_0) + f'(x)(x - x_0) + \frac{1}{2}f''(c)(x - x_0)^2$$

for all $x \in [a, b]$, where c is between x and x_0 . The first two terms represent the tangent line to f , drawn at $(x_0, f(x_0))$. Now, define

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

Note that this is the point at which the tangent line to f at x_0 cuts the x -axis. We have implicitly assumed that $f'(x_0) \neq 0$. In this manner, create the sequence of points

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

We wish to show that $x_n \rightarrow \alpha$, under certain circumstances.

Definition 2.1 (Order of convergence). Let $x_n \rightarrow \alpha$. We say that this convergence is of order $p \geq 1$ if

$$\lim_{n \rightarrow \infty} \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^p} > 0.$$

Theorem 2.2. Let f be a real function on $[\alpha - \delta, \alpha + \delta]$ such that

1. $f(\alpha) = 0$.
2. f is twice differentiable, with non-zero derivatives.
3. f'' is continuous.
4. $|f''(x)/f'(y)| \leq M$ for all x, y .

If $x_0 \in [\alpha - h, \alpha + h]$ where $h = \min\{\delta, 1/M\}$, then the Newton-Raphson sequence generated by x_0 converges to the root α quadratically.

Proof. Pick $x_n \in [\alpha - h, \alpha + h]$. Using Taylor's theorem,

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(c)(\alpha - x_n)^2.$$

Also note that $f(\alpha) = 0$, and $x_n - x_{n+1} = f(x_n)/f'(x_n)$. Thus, dividing by $f'(x_n)$ and substituting gives

$$\alpha - x_{n+1} = -\frac{1}{2} \frac{f''(c)}{f'(x_n)} (\alpha - x_n)^2.$$

Using our estimates on $f''(c)/f'(x_n)$ and x_n along with $h \leq 1/M$, we see that

$$|\alpha - x_{n+1}| \leq \frac{1}{2} M h |\alpha - x_n| \leq \frac{1}{2} |\alpha - x_n|.$$

Indeed, we have shown that

$$|\alpha - x_n| \leq \frac{1}{2^n} |\alpha - x_0|,$$

which directly gives the convergence $x_n \rightarrow \alpha$. Furthermore, we have

$$\frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^2} = \frac{1}{2} \left| \frac{f''(c)}{f'(x_n)} \right| \leq \frac{1}{2} M,$$

hence taking the limit $n \rightarrow \infty$ proves that the convergence is quadratic. \square

Corollary 2.2.1. *Suppose that f satisfies the conditions of the previous theorem, along with $f' > 0$ and $f'' > 0$ on some interval $[\alpha, x]$. Then, the Newton-Raphson sequence generated by $x_0 \in [\alpha, x]$ converges to the root α quadratically.*

Remark. The convexity of f means that the tangent drawn at x_n lies below the curve, and hence cuts the x -axis between α and x_n .

Theorem 2.3. *If α is a multiple root of f such that $f(\alpha) = 0$, $f'(\alpha) = 0$, $f''(\alpha) \neq 0$, then the Newton-Raphson sequence converges to α linearly under suitable conditions.*

Proof. Use Rolle's Theorem to replace $f'(x_n) = f'(x_n) - f'(\alpha) = f''(a)(x_n - \alpha)$. \square

2.4 Secant method

2.5 Fixed point method