

What does this mean for computation? In nearly all our computations, we will replace exact formulations with approximations. For example, we approximate continuous quantities with discrete quantities, we are limited in size by floating point representation of numbers which often necessitates rounding or truncation and introduces (hopefully) small errors. If we are not careful, these error will be amplified within an ill-conditioned problem and produce inaccurate results.

This leads us to consider the *stability* and *accuracy* of our algorithms.

An algorithm is *stable* if the result is relatively unaffected by perturbations during computation. This is similar to the idea of conditioning of problems.

An algorithm is *accurate* if the computed solution is close to the true solution of the problem. Note that a stable algorithm could be inaccurate.

We seek to design and apply stable and accurate algorithms to find accurate solutions to well-posed problems (or to find ways of transforming or approximating ill-posed problems by well-posed ones).

### 3 Approximating a function by a Taylor series

First, a little notation. A real-valued function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is a function  $f$  that takes a real number argument,  $x \in \mathbb{R}$ , and returns a real number,  $f(x) \in \mathbb{R}$ . We write  $f'(x)$  to denote the derivative of  $f$ , so  $f'(x) = \frac{df}{dx}$ , and  $f''(x)$  is the second derivative (so  $f'(x) = \frac{d}{dx}(\frac{df}{dx}) = \frac{d^2f}{dx^2}$ ) and  $f^{(k)}(x)$  is the  $k$ th derivative of  $f$  evaluated at  $x$ .

As we have just seen, simply evaluating a real valued function can be prone to instability (if the function has a large condition number). For that reason, approximations of functions are often used. The most common method of approximating the real-valued function  $f : \mathbb{R} \rightarrow \mathbb{R}$  by a simpler function is to use the Taylor series representation for  $f$ .

The Taylor series has the form of a polynomial where the coefficients of the polynomial are the derivatives of  $f$  evaluated at a point. So long as all derivatives of the function exists at the point  $x = a$ ,  $f(x)$  can be expressed in terms of the value of the function and it's derivatives at  $a$  as:

$$f(x) = f(a) + (x - a)f'(a) + \frac{(x - a)^2}{2!}f''(a) + \dots + \frac{(x - a)^k}{k!}f^{(k)}(a) + \dots$$

This can be written more compactly as

$$f(x) = \sum_{k=0}^{\infty} \frac{(x - a)^k}{k!} f^{(k)}(a),$$

where  $f^{(0)} = f$  and  $0! = 1$  by definition.

This is known as the *Taylor series* for  $f$  about  $a$ . It is valid for  $x$  “close” to  $a$  (strictly, within the “radius of convergence” of the series). When  $a = 0$ , the Taylor series is known as a *Maclaurin series*.

This is an infinite series (the sum contains infinitely many terms) so cannot be directly computed. In practice, we truncate the series after  $n$  terms to get the *Taylor polynomial of degree  $n$*  centred at  $a$ , which we denote  $\hat{f}_n(x; a)$ :

$$f(x) \approx \hat{f}_n(x; a) = \sum_{k=0}^n \frac{(x-a)^k}{k!} f^{(k)}(a).$$

This is an approximation of  $f$  that can be readily calculated so long as the first  $n$  derivatives of  $f$  evaluated at  $a$  can be calculated. The approximation can be made arbitrarily accurate by increasing  $n$ . The quality of the approximation also depends on the distance of  $x$  from  $a$  — the closer  $x$  is to  $a$ , the better the approximation.

**Example:** Find the Taylor approximation of  $f(x) = \exp(x) = e^x$  for values of  $x$  close to 0.

**Solution:** The  $k$ th derivative of  $f(x) = e^x$  is simply  $e^x$  for all  $k$ . Since we want values of  $x$  close to 0, find the Taylor series about  $a = 0$  (the Maclaurin series). Then

$$\hat{f}_n(x; 0) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{k=0}^n \frac{x^k}{k!}$$

This series converges to  $e^x$  everywhere:  $\lim_{n \rightarrow \infty} \hat{f}_n(x) = e^x$ . The quality of the approximation for various values of  $n$  and  $x$  are studied in the table below.

$n$	1	2	3	4	...	true value $e^x$
$\hat{f}_n(x = 1; 0)$	2.0000	2.5000	2.6667	2.7083	...	2.7183
Relative error	0.26	0.08	0.019	0.0037		
$\hat{f}_n(x = 2; 0)$	3.0000	5.0000	6.3333	7.0000	...	7.3891
Relative error	0.59	0.32	0.14	0.053		

Notice that the error is smaller for  $x$  close to  $a$  and decreases as  $n$ , the number of terms in the polynomial increases. □

## 4 Finding the roots of equations

Given a real valued function  $f : \mathbb{R} \rightarrow \mathbb{R}$ , a fundamental problem is to find the values of  $x$  for which  $f(x) = 0$ . This is known as finding the roots of  $f$ . The problem crops up again and again and many problems can be reformulated as this problem. For example, if the trajectory of one object is described by  $h(x)$ , while another object has trajectory  $g(x)$ , then the two objects intercept one another exactly when  $f(x) = g(x) - h(x) = 0$ .

Another common application is when we wish to find the minimum or maximum value that a function takes. We know from basic calculus that if  $f$  is the derivative of some function  $F$ , then  $F(x)$  takes its maximum or minimum values when  $f(x) = 0$ . Thus finding the maximum value of  $F$  is a matter of finding the roots of  $f$ .

We will consider two simple yet effective root finding algorithms. The bisection method and Newton's method. In both cases, we assume that  $f : \mathbb{R} \rightarrow \mathbb{R}$  is continuous.

#### 4.1 Bisection method

We want to find  $x^*$  such that  $f(x^*) = 0$ . Let  $\text{sign}(f(x)) = -1$  if  $f(x) < 0$  and  $\text{sign}(f(x)) = 1$  if  $f(x) > 0$ . The bisection method proceeds as follows:

- **Initialise:** Choose initial guesses  $a, b$  such that  $\text{sign}f(a) \neq \text{sign}f(b)$
- **Iterate** until the absolute difference  $|f(a) - f(b)| \approx 0$ 
  - Calculate  $c = \frac{a+b}{2}$
  - If  $\text{sign}f(a) = \text{sign}f(c)$ , then  $a \leftarrow c$ ; otherwise  $b \leftarrow c$

The method provides slow but sure convergence.

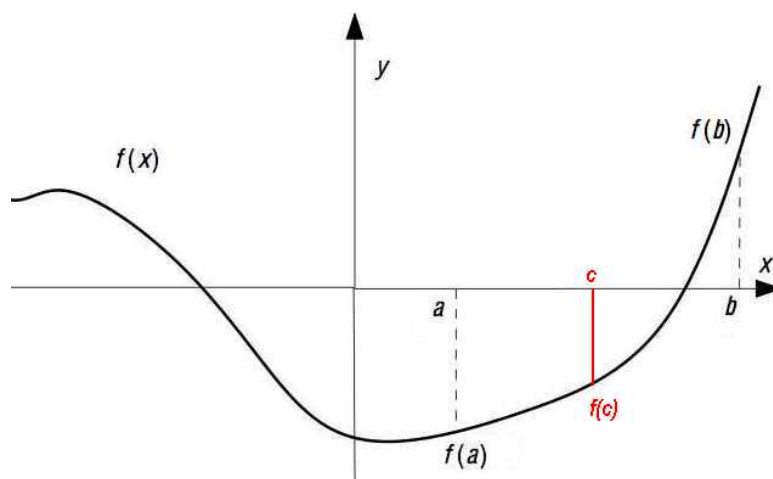


Figure 1: The idea behind the bisection method. Here,  $\text{sign}f(a) = \text{sign}f(c)$  so at the next iteration we will set  $a \leftarrow c$ . The algorithm stops when  $f(a)$  and  $f(b)$  are sufficiently close to each other.

#### 4.2 Newton's method

This method is also known as the Newton-Raphson method and is based on the approximation first two terms of the Taylor series expansion. Recall that we want to find  $x$  such that  $f(x) = 0$ . From the first two terms of the Taylor series of  $f(x)$ , we know that  $f(x) \approx f(a) + (x - a)f'(a)$ . If  $f(x)$  is zero, then the expression on the right hand side to 0 and solve for  $x$  to get

$$f(a) + (x - a)f'(a) = 0 \implies x = a - \frac{f(a)}{f'(a)}.$$