Chapter 1. Introduction

Section 1. Overview

1. Applications of numerical methods.(Burden & Faires, 1.1)

As the development of the modern computer technology, numerical approximations are widely used in today's real world problems. In general, when the analytic form of the solution to a problem is not known, or the analytic form is too complicated to be usable, then the numerical approximation is used to find an approximate solution to the problem. The following are some examples of numerical approximations:

- Using Taylor's theorem we have

$$\cos x = 1 - \frac{1}{2}x^2 + \frac{1}{6}x^3 \sin \xi$$

where ξ is some number between 0 and x. When $x = 0.01$, this becomes

$$\cos(0.01) = 1 - \frac{1}{2}(0.01)^2 + \frac{1}{6}(0.01)^3 \sin \xi = 0.99995 + \frac{10^{-6}}{6} \sin \xi$$

The approximation to $\cos(0.01)$ is 0.99995, the truncation error is

$$\frac{10^{-6}}{6} \sin \xi = 0.1\bar{6} \times 10^{-6} \sin \xi$$

Therefore,

$$|\cos(0.01) - 0.99995| = 0.1\bar{6} \times 10^{-6} \sin \xi \leq 0.1\bar{6} \times 10^{-6}$$

- In many application problems, it is impossible to find the analytic form of the integral

$$\int_a^b f(x)dx$$

and in some cases, we do not even know the expression of $f(x)$, but only know the function values for any given x. For this kind of integrals, numerical integration is necessary. A simple approximation formula is

$$\int_a^b f(x)dx \approx \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

If $f(x)$ is integrable, then this approximation will converge to the integral, i.e.

$$\lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(x_i) = \int_a^b f(x)dx$$

- The solution to the following initial boundary value problem

$$\begin{aligned} &\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad x > 0, t > 0, \\ &u(0,t) = 0 \qquad t > 0, \\ &u(x,0) = f(x) \qquad x > 0 \end{aligned}$$

  is given by

$$u(x,t) = \frac{2}{\pi} \int_0^\infty \int_0^\infty f(\xi) e^{-\omega^2 t} \sin(\omega \xi) \sin(\omega x) d\xi d\omega$$

  If the function $f(x)$ is not simple, then the right hand side can not be integrated. For example,

$$f(x) = e^{-x^2} \sin x^2$$

  If the function value $u(x,t)$ is needed for given x and t, then numerical integration must be used to evaluate the integral.

- In general, when the equation is nonlinear, it is difficult to find the analytic expression of the solution. For example, consider the initial value problem

$$\begin{aligned} &\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + e^u \qquad -\infty < x < \infty, t > 0, \\ &u(x,0) = f(x) \end{aligned}$$

  If the value $u(1, 0.01)$ is needed, we can use numerical differentiation to approximate the derivatives,

$$\begin{aligned} \frac{\partial u}{\partial t} &\approx \frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} \\ \frac{\partial^2 u}{\partial x^2} &\approx \frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} \end{aligned}$$

  Then we have the approximate equation

$$\frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} \approx \frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} + e^{u(x,t)}$$

  Therefore,

$$u(x, t+\Delta t) \approx (1 - 2\lambda) u(x,t) + \lambda (u(x+\Delta x, t) + u(x-\Delta x, t)) + \Delta t e^{u(x,t)}$$

  where $\lambda = \Delta t / (\Delta x)^2$. Let $x = 1$, $t = 0$, $\Delta t = 0.01$, $\Delta x = 0.2$, and use the initial condition, we can obtain an approximation value of $u(1, 0.01)$.

- In solving the real world problems, very often the last step is to find the solution of a linear system of equations,

$$A\mathbf{x} = \mathbf{b}$$

  When the matrix A is large, it is usually impossible to find the exact solution $\mathbf{x}$. Therefore, numerical methods are used to find an approximate solution to the system.

The following are some examples of the applications of the numerical methods:

- Design of aircraft, bridges, and skyscrapers
- Whether forecast
- Design of electronic circuits
- Signal and image processing
- Pricing of options and futures
- Inventory and scheduling optimization

2. Computational errors. (Burden & Faires, 1.2)

As we have seen in the previous examples, when an numerical approximation is applied, we introduce errors. There are two kinds of errors.

Truncation error. Consider the Taylor's theorem,

$$f(x) = P_n(x) + R_n(x)$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{(n+1)}$$

If we use $P_n(x)$ to approximate $f(x)$, then the error is $R_n(x)$. This error is called the truncation error.

Round-off error. In the numerical computation by computers, the numbers are represented in the **decimal floating-point form**,

$$\pm 0.d_1 d_2 \cdots d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad 0 \le d_i \le 9, \quad i = 2, \cdots, k$$

Numbers of this form are called k**-digit decimal machine numbers**. A single precision number has 7 to 8 digits, and a double precision number has 15 to 16 digits. Any positive real number has the form

$$y = 0.d_1 d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \times 10^n.$$

There are two ways to approximate y in computers. One is called **chopping**, we simply chop off the digits $d_{k+1} d_{k+2} \cdots$. This produces the floating-point form

$$fl(y) = 0.d_1 d_2 \cdots d_k \times 10^n.$$

The other one is called **rounding**, we add $5 \times 10^{n-(k+1)}$ to y and then chop the result to obtain

$$fl(y) = 0.\delta_1\delta_2 \cdots \delta_k \times 10^n,$$

or equivalently, if $d_{k+1} \geq 5$, we add 1 to d_k to obtain $fl(y)$, i.e., we round up. If $d_{k+1} < 5$, we just chop off all but the first k digits, so we round down.

To measure the errors, we have the following two ways.

Absolute and relative error. If p^* is an approximation to p, then $|p - p^*|$ is called the absolute error, and $|p - p^*|/|p|$ is called the relative error.

Significant digits. The number p^* is said to approximate p to t significant digits if t is the largest nonnegative integer for which

$$\frac{|p - p^*|}{|p|} \leq 5 \times 10^{-t}$$

The following table shows that p^* approximate p to 4 significant digits.

p	0.1	0.5	100	1000	5000	9990	10000
$\max \lvert p - p^* \rvert$	0.00005	0.00025	0.05	0.5	2.5	4.995	5

One of the most common error producing calculations involves the cancelation of significant digits due to subtraction of nearly equal numbers. Let

$$fl(x) = 0.d_1d_2 \cdots d_p\alpha_{p+1}\alpha_{p+2} \cdots \alpha_k \times 10^n$$
$$fl(y) = 0.d_1d_2 \cdots d_p\beta_{p+1}\beta_{p+2} \cdots \beta_k \times 10^n$$

If $x > y$, then

$$fl(fl(x) - fl(y)) = 0.\sigma_{p+1}\sigma_{p+2} \cdots \sigma_k \times 10^{n-p}$$

where

$$0.\sigma_{p+1}\sigma_{p+2} \cdots \sigma_k = 0.\alpha_{p+1}\alpha_{p+2} \cdots \alpha_k - 0.\beta_{p+1}\beta_{p+2} \cdots \beta_k$$

The floating number used to represent $z = x - y$ has at most $k - p$ significant digits, the last p digits are random numbers. Thus, an error δ is introduced in the calculation. If $x - y$ is further divided by a small number or multiplied by a large number, then the error will be enlarged. If $\epsilon = 10^{-n}$ with $n > 0$, then

$$\frac{z}{\epsilon} \approx fl\left(\frac{fl(z)}{fl(\epsilon)}\right) = (z + \delta) \times 10^n$$

Thus, the absolute error is $|\delta| \times 10^n$, i.e., the original error δ is enlarged by a factor 10^n.

Example. The two roots of $ax^2 + bx + c = 0$ are given by

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Consider the equation $x^2 + 62.10x + 1 = 0$. The two roots are

$$x_1 = -0.01610723, \quad x_2 = -62.08390$$

In this equation, b^2 is much larger than $4ac$, so the numerator of x_1 is a subtraction of two nearly equal numbers. Use 4-digit rounding we get

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = -0.02000$$

which is a poor approximation to x_1 with the large relative error

$$\frac{|-0.01611 + 0.02000|}{|-0.01611|} \approx 2.4 \times 10^{-1}$$

To improve the accuracy of the calculation, we change the formula by **rationalizing the numerator**,

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}\right) = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

Using this we get

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610$$

which has the small relative error 6.2×10^{-4}.

Example. Evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ for $x = 4.71$. Using 3 digit arithmetic we have,

	x	x^2	x^3	$6.1x^2$	$3.2x$
Exact	4.71	22.1841	104.487111	135.32301	15.072
Chopping	4.71	22.1	104.	134.	15.0
Rounding	4.71	22.2	105.	135.	15.1

The exact value of $f(4.71)$ is

$$f(4.71) = 104.487111 - 135.32301 + 15.072 + 1.5 = -14.263899$$

Using chopping we have

$$f(4.71) = 104. - 134. + 15.0 + 1.5 = -13.5$$

The relative error is

$$\left|\frac{-14.263899 + 13.5}{-14.263899}\right| \approx 0.05$$

Using rounding we have

$$f(4.71) = 105. - 135. + 15.1 + 1.5 = -13.4$$

The relative error is

$$\left|\frac{-14.263899 + 13.4}{-14.263899}\right| \approx 0.06$$

In both cases, large relative errors occur. The reason is that large numbers are produced due to the exponents, then subtraction of nearly equal numbers causes the loss of significant digits. To improve the calculation, we change the polynomial to the following **nested** form,

$$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5 = ((x - 6.1)x + 3.2)x + 1.5$$

Then using chopping we have

$$f(4.71) = ((4.71 - 6.1)4.71 + 3.2)4.71 + 1.5 = -14.2$$

The relative error is

$$\left|\frac{-14.263899 + 14.2}{-14.263899}\right| \approx 0.0045$$

Using rounding we have

$$f(4.71) = ((4.71 - 6.1)4.71 + 3.2)4.71 + 1.5 = -14.3$$

The relative error is

$$\left|\frac{-14.263899 + 14.3}{-14.263899}\right| \approx 0.0025$$

3. Convergence and stability.(Burden & Faires, 1.3)

Numerical methods usually involve approximation sequences, iteration techniques, etc. The convergence of the numerical methods is a very important concept in approximations. Also, the rate of convergence is a measurement to different algorithms.

Convergence. Suppose $\{\beta_n\}_{n=1}^{\infty}$ is a sequence known to converge to zero, and $\{\alpha_n\}_{n=1}^{\infty}$ converges to a number α. If a positive constant K exists with

$$|\alpha_n - \alpha| \leq K|\beta_n| \quad \text{for large } n$$

then we say that $\{\alpha_n\}_{n=1}^{\infty}$ converges to α with the rate of convergence $O(\beta_n)$ (big oh of β_n), and written as

$$\alpha_n = \alpha + O(\beta_n)$$

In most cases, $\beta_n = 1/n^p$ $(p > 0)$, so we have

$$\alpha_n = \alpha + O\left(\frac{1}{n^p}\right)$$

Similarly, if a function $F(h)$ satisfies

$$|F(h) - L| \leq Kh^p \quad \text{for small positive } h$$

we write

$$F(h) = L + O(h^p)$$

In general, a computer algorithm involves many steps of computation. Suppose, for example, a error is introduced in the first step of computation. As we have seen in the previous examples, this error may be enlarged in the subsequent computations, and the final result may be totally wrong dut to the large error. The stability concept is related to this issue.

Stability. If small changes in the initial data produce correspondingly small changes in the final results, then we say that the algorithm is stable. Otherwise, the algorithm is said unstable.

Example. The solution of the recursive equation

$$p_n = \frac{10}{3}p_{n-1} - p_{n-2}, \quad n = 2, 3, \cdots$$

is

$$p_n = c_1 \left(\frac{1}{3}\right)^n + c_2 3^n$$

If $p_0 = 1$ and $p_1 = 1/3$, we have $c_1 = 1$ and $c_2 = 0$, then $p_n = (1/3)^n$. If we use 5 digit rounding to compute p_n, then the computed sequence $\hat{p}_0 = 1.0000$, $\hat{p}_1 = 0.33333$, which corresponding to $\hat{c}_1 = 1.0000$ and $\hat{c}_2 = -0.12500 \times 10^{-5}$, and

$$\hat{p}_n = 1.0000 \left(\frac{1}{3}\right)^n - 0.12500 \times 10^{-5}(3^n)$$

The round-off error is

$$p_n - \hat{p}_n = 0.12500 \times 10^{-5}(3^n)$$

This error grows expontially, so the algorithm is unstable.

Example. The solution of the recursive equation

$$p_n = \frac{3}{4}p_{n-1} - \frac{1}{8}p_{n-2}, \quad n = 2, 3, \cdots$$

is

$$p_n = c_1 \left(\frac{1}{2}\right)^n + c_2 \left(\frac{1}{4}\right)^n$$

If the computed values for c_1 and c_2 are $\hat{c}_1$ and $\hat{c}_2$, due to round-off errors, then we have

$$\hat{p}_n = \hat{c}_1 \left(\frac{1}{2}\right)^n + \hat{c}_2 \left(\frac{1}{4}\right)^n$$

Thus, the round-off error in computing p_n is

$$\begin{aligned} |p_n - \hat{p}_n| &= \left| (c_1 - \hat{c}_1)\left(\frac{1}{2}\right)^n + (c_2 - \hat{c}_2)\left(\frac{1}{4}\right)^n \right| \\ &\leq |c_1 - \hat{c}_1|\left(\frac{1}{2}\right)^n + |c_2 - \hat{c}_2|\left(\frac{1}{4}\right)^n \\ &\leq |c_1 - \hat{c}_1|\frac{1}{4} + |c_2 - \hat{c}_2|\frac{1}{16} \end{aligned}$$

Small changes in the initial errors $|c_1 - \hat{c}_1|$ and $|c_2 - \hat{c}_2|$ will result in a small change in $|p_n - \hat{p}_n|$, the algorithm is stable.

Relation between convergence and stability. The concept of convergence is about the theoretical errors (truncation errors) between the true value and the exact value of an algorithm. The concept of stability is about the computional errors (round-off error) between the true value of an algorithm and the computated value from the algorithm. Take the previous example, consider the algorithm

$$p_n = \frac{10}{3}p_{n-1} - p_{n-2}, \quad n = 2, 3, \cdots$$

When $p_0 = 1$ and $p_1 = 1/3$, we have $p_n = (1/3)^n$. Clearly,

$$\lim_{n\to\infty} p_n = 0$$

However, for the computed value

$$\lim_{n\to\infty} \hat{p}_n = \infty$$