МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Нижегородский государственный университет им. Н.И. Лобачевского

К.А.Баркалов

Численные методы

Учебно-методическое пособие

Рекомендовано методической комиссией факультета ВМК для иностранных студентов, обучающихся в ННГУ по направлению подготовки 010300 «Фундаментальная информатика и информационные технологии» (бакалавриат)

1-е издание

Нижний Новгород

2011

Баркалов К.А. Численные методы: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2011. – 38 с.

В настоящем пособии изложены учебно-методические материалы по курсу «Численные методы» для иностранных студентов, обучающихся в ННГУ по направлению подготовки 010300 «Фундаментальная информатика и информационные технологии» (бакалавриат).

Ministry of Education and Science of the Russian Federation

State educational institution of higher education
«Lobachevsky State University of Nizhni Novgorod»

К.А.БАркалов

Computing mathematics

Tutorial

Recommended by the Methodical Commission of the Faculty of Computer Science for international students, studying at the B.Sc. program 010300 "Fundamental Informatics and Information Technologies"

Nizhny Novgorod

2011

Introduction

Computing mathematics is the study of algorithms that use numerical approximation (as opposed to general symbolic manipulations) for the problems of mathematical analysis and algebra (as distinguished from discrete mathematics).

Computing mathematics naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century, the life sciences and even the arts have adopted elements of scientific computations. Ordinary differential equations appear in the movement of heavenly bodies (planets, stars and galaxies); optimization occurs in portfolio management; numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology.

Before the advent of modern computers numerical methods often depended on hand interpolation in large printed tables. Since the mid 20th century, computers calculate the required functions instead. The interpolation algorithms nevertheless may be used as part of the software for solving differential equations.

The field of numerical analysis predates the invention of modern computers by many centuries. Linear interpolation was already in use more than 2000 years ago. Many great mathematicians of the past were preoccupied by numerical analysis, as is obvious from the names of important algorithms like Newton's method, Lagrange interpolation polynomial, Gaussian elimination, or Euler's method.

To facilitate computations by hand, large books were produced with formulas and tables of data such as interpolation points and function coefficients. Using these tables, often calculated out to 16 decimal places or more for some functions, one could look up values to plug into the formulas given and achieve very good numerical estimates of some functions. The function values are no longer very useful when a computer is available, but the large listing of formulas can still be very handy.

The mechanical calculator was also developed as a tool for hand computation. These calculators evolved into electronic computers in the 1940s, and it was then found that these computers were also useful for administrative purposes. But the invention of the computer also influenced the field of numerical analysis, since now longer and more complicated calculations could be done.

Course program

1. Finite-digit arithmetic
    - 1.1. Positional (or radix) notation
    - 1.2. Normalized scientific notation
    - 1.3. Machine epsilon
    - 1.4. Absolute and relative errors
2. Solution of nonlinear equations
    - 2.1. Bisection method
        - 2.1.1. Error analysis

1. Finite-digit arithmetic

1.1. Positional (or radix) notation

Consider different notations of one number.

Decimal system: $49.75 = 4\cdot10+9\cdot1+7\cdot0.1+5\cdot0.01 = 4\cdot10^{1}+9\cdot10^{0}+7\cdot10^{-1}+5\cdot10^{-2}$

Binary system: $11001.11 =1\cdot2^{5}+1\cdot2^{4}+0\cdot2^{3}+0\cdot2^{2}+0\cdot2^{1}+1\cdot2^{0}+1\cdot2^{-1}+1\cdot2^{-2}$

Theorem (without proof). For any integer $p\geq2$, any real number x may be represented in the form

$$x = \beta_k p^k + \beta_{k-1} p^{k-1} + \ldots \beta_1 p^1 + \beta_0 p^0 + \beta_{-1} p^{-1} + \ldots + \beta_{-n} p^{-n} \qquad (1.1)$$

where coefficients β_i are integer and satisfy the inequalities $0 \le \beta_i \le p-1$. □

Let's reduce the notation (1.1) and write the number **X** as the sequence of coefficients β_i:

$$x = \beta_k \beta_{k-1} \beta_1 \beta_0 . \beta_{-1} \beta_{-n} \tag{1.2}$$

The reduced form (1.2) is called positional representation of the number x in the number system notation with radix p.

1.2. Normalized scientific notation

In the decimal system, any real number can be expressed in normalized scientific notation. This means that the decimal point is shifted and appropriate powers of 10 are supplied so that all the digits are to the right of the decimal point, and the first digit displayed is not zero.

$$x = \pm r \cdot 10^n \tag{1.3}$$

where $1/10 \le r < 1$, and n is integer.

In exactly the same way, we can use scientific notation in the binary system

$$x = \pm q \cdot 10^m \tag{1.3}$$

where $1/2 \le r < 1$, and m is integer. The number q is called the ***mantissa*** and the integer m the ***exponent***; both q and m will be represented as base 2 numbers.

1.3. Machine epsilon

If we store a real number in 32-bit memory (4 bytes) and use 1 bit for the number sign, 7 bits for the exponent part and 24 bits for the representation of the floating point coefficient then we can represent the numbers with 6-7 significant digits in the interval from 10^{-38} to 10^{38} (approximately) for positive numbers and symmetrical interval for negative ones.

Such format for representation of real numbers defines the data type float in the C language.

For more precise computations ensuring more accuracy the type double is used. It requires 8 bytes in memory and extends the quantity of significant digits up to 15-16 and the range of possible positive real numbers from 10^{-308} to 10^{308} (negative numbers are included symmetrically).

If a computer operates with base 2 and carries n places in the mantissa of its floating-point numbers, then

$$\text{float } x = x \pm \delta,\ 0 \le \delta \le \varepsilon = 2^{1-n}.$$

The number ε is the unit roundoff error and is a characteristic of the computer, its operating system, and the mode of computation (whether single- or double-precision).

To compute an approximate value of ε on any machine, the following algorithm can be used, in either single- or double-precision. It determines the smallest positive number ε of the form 2^{-k} such that $1.0 + \varepsilon \neq 1.0$ in the machine.

Algorithm.

Step 1. eps = 1.0; k = 0;
Step 2. eps = eps/2;
Step 3. k = k+1;
Step 4. If (1+eps)=1 then STOP;
else GOTO Step 2;
Step 5. Output eps;

1.4. Absolute and relative errors

In the practice of numerical analysis it is important to be aware that computed solutions are not exact mathematical solutions. The precision of a numerical solution can be diminished in several ways.

Definition. Suppose that p_1 is an approximation to p. The *absolute error* is

$$E_p=|p - p_1|,$$

and the *relative error* is

$$R_p= |p - p_1|/|p|,$$

provided that p≠0.

The error is simply the difference between the true value and the approximate value, whereas the relative error is a portion of the true value.

Example. Find the error and relative error in the following three cases.

a. Let x=3.141592 and x_1=3.14. Then

$$E_x=|x - x_1|=|3.141592 - 3.14|=0.001592,$$
$$R_x=|x - x_1|/|x|= 0.001592 / 3.141592 =0.00507.$$

b. Let y=100000 and y1=999996. Then

$$E_y=|y - y_1|=|1000000 - 9999996|=4,$$
$$R_y=|y - y_1|/|y|= 4 / 1000000 =0.000004.$$

c. Let z=0.000012 and z1=0.000009. Then

$$E_z=|z - z_1|=|0.000012 - 0.000009|= 0.000003$$
$$R_z=|z - z_1|/|z|= 0.000003 / 0.000012 =0.25.$$

If $|p|$ moves away from 1 (greater than or less than) the relative error R_p is a better indicator of the accuracy of the approximation than E_p. Relative error is preferred for floating-point representations since it deals directly with the mantissa.

2. Solution of nonlinear equations

This part is devoted to the problem of determining roots of equations (or zeros of functions). It is a problem of frequent occurrence in scientific work. For example, in the theory of diffraction of light, we need the roots of the equation

$$x - \tan x = 0.$$

In the calculation of planetary orbits, we need the roots of Kepler's equation for various values of a and b

$$x - a \sin x = b.$$

The general question, posed in the simplest case of a real-valued function of a real variable, is this: given a real-valued function $f(x)$, find the values of x for which $f(x)=0$. We shall consider several of the standard procedures for solving this problem.

2.1. Bisection method

If $f(x)$ is a continuous function on the interval $[a,b]$ and if $f(a)\cdot f(b)<0$, then $f(x)$ must have at least one zero in the interval (a,b). The bisection method exploits this idea in the following way. If f(a)f(b)<0 then we compute

$$c = (a + b)/2$$

and test whether $f(a)\cdot f(c)<0$. If so, then $f(x)$ has a zero in $[a,c]$. So we rename c as b and start again with the new interval $[a,b]$, which is half as large as the original interval. If $f(a)\cdot f(c)>0$ then $f(b)\cdot f(c)<0$, and in this case we rename c as a. In either case, a new interval containing a zero of $f(x)$ has been produced, and the process can be repeated.

If $f(a)\cdot f(c)=0$ then $f(c)=0$ and a zero has been found. However, it is quite unlikely that $f(c)$ will be zero in the computer because of roundoff errors. Thus, the stopping criterion should not be whether $f(c)=0$.

A reasonable tolerance must be allowed, such as

$$|f(c)| < \varepsilon$$

or

$$b-a < \delta,$$

where ε and δ is rather small.

To illustrate the process of root finding we consider two different cases. In the first case (figure 1) bisection method selects left subinterval.

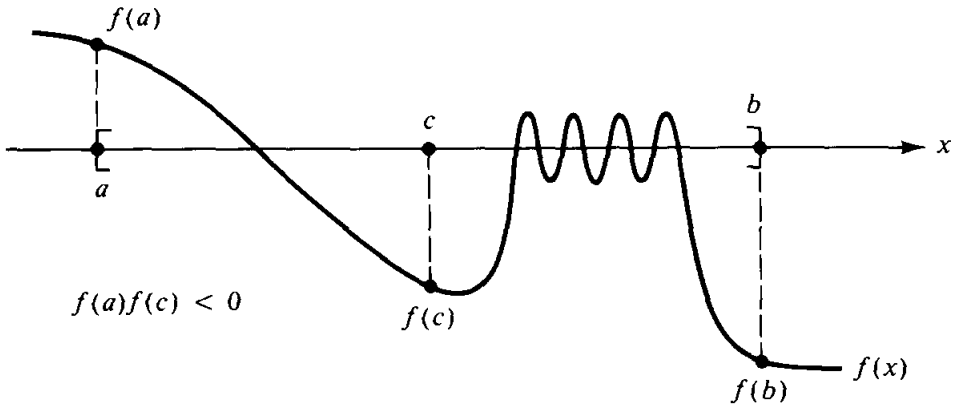


Figure 1. Bisection method selects left subinterval

In the second case (figure 2) bisection method selects right subinterval.

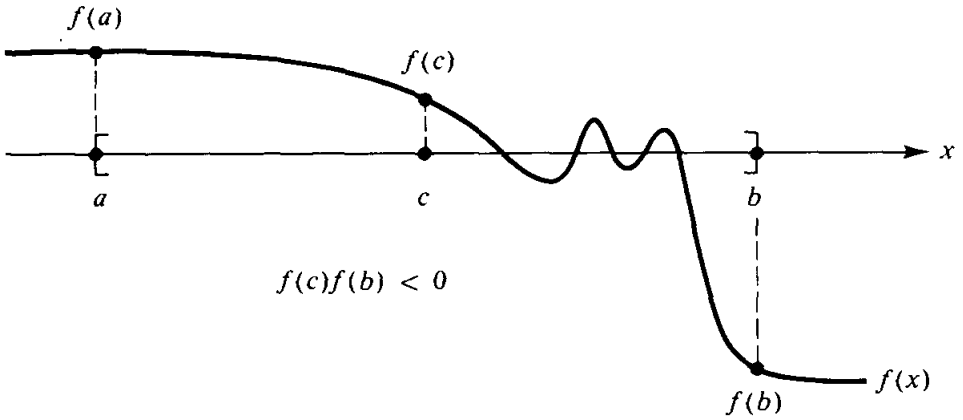


Figure 2. Bisection method selects left subinterval

On figure 3 we illustrate three iterations of bisection method.

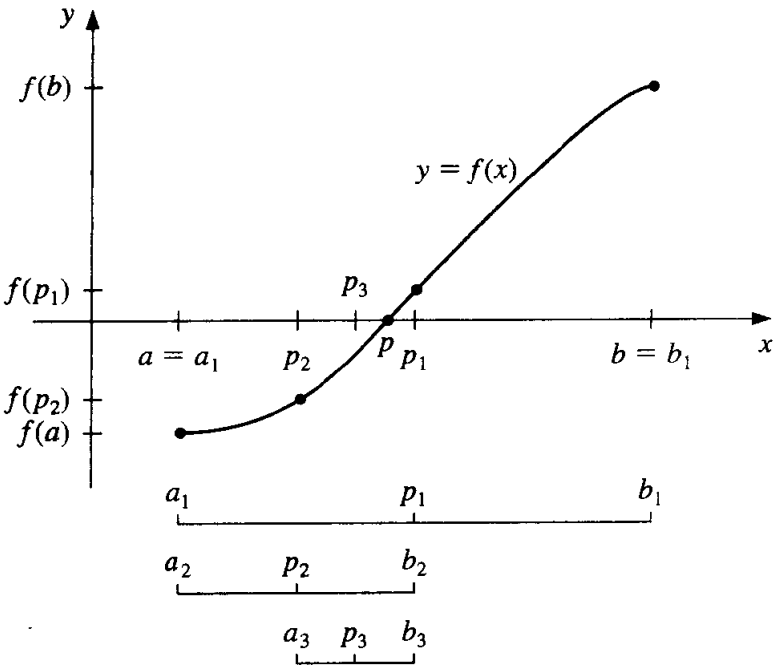


Figure 3. Iterations of bisection method

Now we are ready to write the algorithm. Input of the algorithm are endpoints *a* and *b*; tolerance TOL; maximum number of iterations *N*. Output of the algorithm are solution approximation *c* or message of failure.

```
Step 1. Set i = 1; Fa = f (a).
Step 2  While i < N do Steps 3-6.
Step 3  Set c= a + (b-a)/2; //Compute c
          Fc = f(c).
Step 4  If Fc = 0 or (b - a)/2 < TOL then
            OUTPUT (c); //Procedure completed successfully
            STOP.
Step 5  Set i = i + 1.
Step 6 If Fa-Fc > 0 then
            set a = c;
            Fa = Fc
        else
            set b = c.
Step 7 OUTPUT ('Method failed after N iterations, N =', N);
         //The procedure was unsuccessful.
         STOP.
```

2.1.1. Error analysis

To analyze the bisection method, let us denote the successive intervals that arise in the process by $[a_0, b_0]$, $[a_1,b_1]$, and so on. Here are some observations about these numbers:

1) $a_0 \le a_1 \le \ldots \le a_n \le b_0$

2) $b_0 \ge b_1 \ge \ldots \ge b_n \ge a_0$

3) $b_{n+1} - a_{n+1} = 0.5\,(b_n - a_n)$

If we apply (3) repeatedly, we find that $b_n - a_n = 2^{-n}\,(b_0 - a_0)$

Thus

$$\lim_{n\to\infty} b_n - \lim_{n\to\infty} a_n = \lim_{n\to\infty} 2^{-n}(b_0 - a_0) = 0$$

If we put

$$r = \lim_{n\to\infty} b_n = \lim_{n\to\infty} a_n$$

then, by taking a limit in the inequality $0 \ge f(a_n)\cdot f(b_n)$, we obtain

$$0 \ge [f(r)]^2,$$

whence

$$f(r) = 0.$$

Suppose that, at a certain stage in the process, the interval $[a_n,b_n]$ has just been defined. If the process is now stopped, the root is certain to lie in this interval. The best estimate of the root at this stage is not a_n or b_n but the midpoint of the interval:

$$c_n = (a_n + b_n)/2$$

The error is then bounded as follows

$$|r - c_n| \leq \frac{1}{2}|b_n - a_n| \leq 2^{-(n+1)}(b_0 - a_0)$$

Theorem. If $[a_0, b_0]$, $[a_1,b_1]$,…, $[a_n,b_n]$… denote the intervals in the bisection method, then the limits $\lim_{n\to\infty} a_n$ and $\lim_{n\to\infty} b_n$ exist, are equal, and represent a zero of *f*. If $r = \lim_{n\to\infty} c_n$ and $c_n = (a_n + b_n)/2$, then

$$|r - c_n| \leq 2^{-(n+1)}(b_0 - a_0).$$

Example. Suppose that the bisection method is started with the interval [50,63]. How many steps should be taken to compute a root with relative accuracy of one part in 10^{-12}?

Solution. The stated requirement on relative precision means that

$$\frac{|r-c_n|}{|r|} \leq 10^{-12}.$$

We know that $r \geq 50$, and thus it will suffice to secure the inequality

$$\frac{|r-c_n|}{50} \leq 10^{-12}.$$

By means of the preceding theorem, we infer that the following condition will be sufficient

$$2^{-(n+1)}\frac{13}{50} \leq 10^{-12}.$$

Solving this for n, we conclude that $n \geq 37$.

2.2. Newton's method

Newton's method is a general procedure that can be applied in many diverse situations. When specialized to the problem of locating a zero of a real-valued function of a real variable, it is often called the *Newton-Raphson iteration*.

We have a function $f(x)$ whose zeros are to be determined numerically. Let r be a zero of $f(x)$ and let x be an approximation to r. If $f''(x)$ exists and is continuous, then by Taylor's theorem

$$0 = f(r) = f(x + h) = f(x) + hf'(x) + O(h^2)$$

where $h=r–x$. If h is small (that is, x is near r), then it is reasonable to ignore the $O(h^2)$-term and solve the remaining equation for h.

Therefore, the result is $h = -f(x)/f'(x)$. If x is an approximation to r, then $x - f(x)/f'(x)$ should be a better approximation to r.

Newton's method begins with an estimate x_0 of r and then defines inductively

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n > 0.$$

The stopping criterion: $|f(r)| < \varepsilon$ or $|x_{n+1}-x_n| < \delta$, where ε and δ are rather small.

2.2.1. Graphical interpretation of Newton's method

From the description already given, we can say that Newton's method involves *linearizing* the function. That is, $f(x)$ was replaced by a linear function. The usual way of doing this is to replace $f(x)$ by the first two terms in its Taylor series. Thus if

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

then the linearization (at c) produces the linear function

$$l(x) = f(c) + f'(x)(x-c).$$

Notice that $l(x)$ is a good approximation to $f(x)$ in the vicinity of c, and in fact we have $l(c) = f(c)$ and $l'(c) = f'(c)$. Thus, the linear function has the same value and the same slope as $f(x)$ at the point c.

So in Newton's method we are constructing the tangent to the f-curve at a point near r, and finding where the tangent line intersects the x-axis (see figure 4).

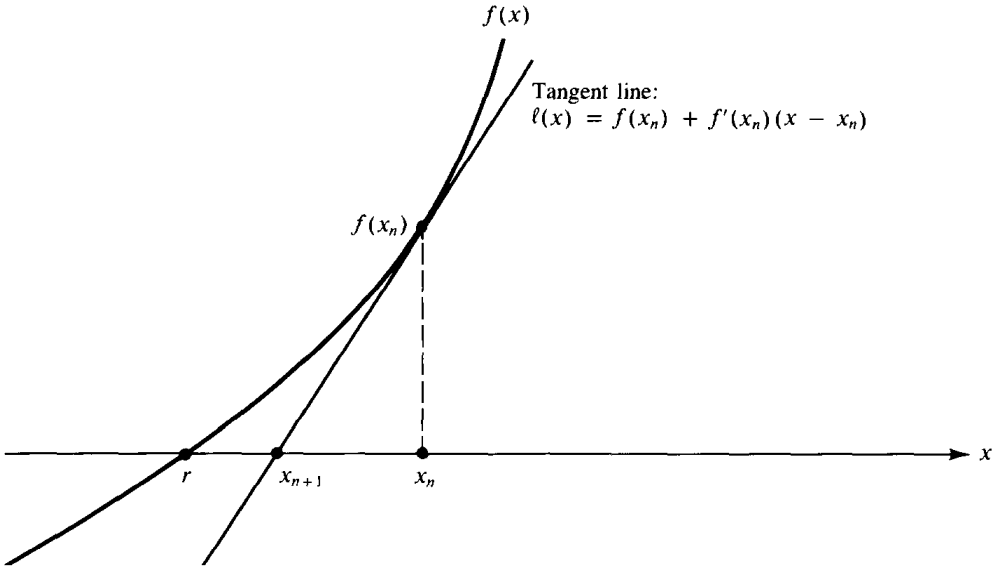


Figure 4. Linearizing the function

Keeping in mind this graphical interpretation, we can easily imagine functions and starting points for which the Newton iteration will fail (see figure 5).

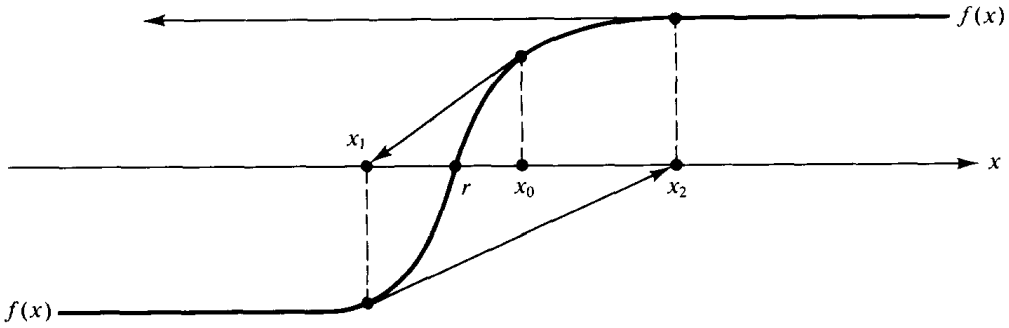


Figure 5. Newton iteration fails

In this example, the shape of the curve is such that for certain starting values, the sequence $[x_n]$ will diverge. Thus, any formal statement about Newton's method must involve an assumption that x_0 is *sufficiently close to a zero*, or that the graph of $f(x)$ has a prescribed shape.

Now we are ready to write the algorithm. Input of the algorithm are initial approximation x_0; tolerance TOL; maximum number of iterations N. Output of the algorithm are solution approximation x or message of failure.

Step 1 Set i = 1.
Step 2 While i < N do Steps 3-6.
Step 3 Set x = x_0 - f(x_0)/f'(x_0). //Compute p
Step 4 If | x – x_0 | < TOL then
    OUTPUT (x) //The procedure was successful.
    STOP.
Step 5 Set i = i + 1.
Step 6 Set x_0 = x // Update x_0.
Step 7 OUTPUT ('The method failed after N iterations, N =', N0);
    //The procedure was unsuccessful.
    STOP.

2.2.2. Error analysis

By errors, we mean the quantities $\boldsymbol{e_n=x_n-r}$. Let us assume that f'' is continuous and r is a simple zero of f, so that $f(r)=0\neq f'(r)$. From the definition of the Newton iteration, we have

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

By Taylor's theorem, we have

$$0 = f(r) = f(x_n - e_n) = f(x_n) - e_n f'(x_n) + \frac{1}{2} e_n^2 f''(\xi_n)$$

where $x_n<\xi_n<r$. From this equation, we have

$$e_n f'(x_n) - f(x_n) = \frac{1}{2} e_n^2 f''(\xi_n)$$

and

$$e_n f'(x_n) - f(x_n) = \frac{1}{2}\frac{f''(\xi_n)}{f'(x_n)} e_n^2 \approx \frac{1}{2}\frac{f''(r)}{f'(r)} e_n^2 = C e_n^2.$$

This equation tells us that $\boldsymbol{e_{n+1}}$ is roughly a constant times $\boldsymbol{e_n}^2$. This desirable state of affairs is called *quadratic convergence*. It accounts for the apparent doubling of precision with each iteration of Newton's method.

Example. Find an efficient method for computing square roots based on the use of Newton's method.

Solution. Let $R>0$, and $x = \sqrt{R}$. Then x is a root of the equation $x^2 - R = 0$. If we use Newton's method on the function $f(x) = x^2 - R$, the iteration formula can be written as

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{R}{x_n}\right).$$

This formula is very ancient, and is credited to Heron, a Greek engineer and architect who lived sometime between 100 B.C. and 100 A.D. If, for example, we wish to compute $\sqrt{17}$ and begin with $x_0 = 4$, the successive approximants are as follows

$$x_1 = 4.12$$
$$x_2 = 4.123\ 106$$
$$x_3 = 4.123\ 1056\ 2561\ 77$$

2.3. Fixed-Point Iteration

Definition. A number p is a fixed point for a given function $g(x)$ if $g(p)=p$.

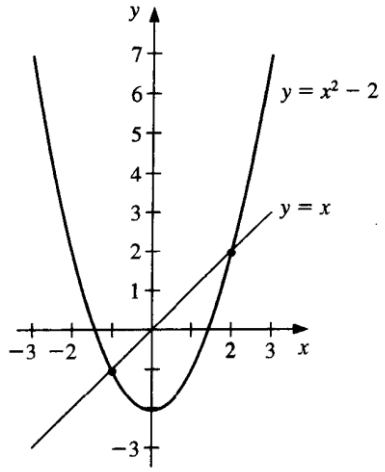


Figure 6. Fixed points

Root-finding problems and fixed-point problems are equivalent classes in the following sense. Given a root-finding problem $f(x)=0$, we can define functions $g(x)$ with a fixed point at p, e.g.,

$$g(x) = x - f(x).$$

Conversely, if the function $g(x)$ has a fixed point at x, then the function $f(x)$ has a zero at p.

Consider as an example function $y(x)=x^2-2$ (see figure 6). There are two fixed points of this function $p_1=2$, $p_2=-1$

The following theorem gives sufficient conditions for the existence and uniqueness of a fixed point.

Theorem. (a) If $g(x)\in C[a,b]$ and $g(x)\in[a,b]$ for all $x\in[a,b]$, then $g(x)$ has a fixed point in $p\in[a,b]$. (b) If, in addition, $g'(x)$ exists on (a,b) and a positive constant $k<1$ exists with $|g'(x)|\le k$, for all $x\in(a,b)$, then the fixed point $p\in[a,b]$ is unique.

Proof. (a) If $g(a)=a$ or $g(b)=b$, then $g(x)$ has a fixed point at an endpoint. If not, then $g(a)>a$ and $g(b)<b$. The function $h(x)=g(x)-x$ is continuous on $[a,b]$, with

$$h(a)=g(a)-a>0 \text{ and } h(b)=g(b)-b < 0.$$

The Intermediate Value Theorem implies that there exists $p\in(a,b)$ for which $h(p)=0$. This number p is a fixed point for $g(x)$ since

$$0=h(p)=g(p)-p$$

implies that $g(p)=p$.

(b) Suppose, in addition, that $|g'(x)|\le k<1$ and that p and q are both fixed points in $[a,b]$. lf $p\ne q$, then the Mean Value Theorem implies that a number ξ exists, $p<\xi<q$, with

$$\frac{g(p) - g(q)}{p - q} = g'(\xi)$$

Thus,

$$|p-q|=|g(p)-g(q)|=|g'(\xi)||p-q|\leq k|p-q|<|p-q|$$

which is a contradiction. This contradiction must come from the only supposition, $p\neq q$. Hence, $p=q$ and the fixed point in $[a,b]$ is unique.

This theorem illustrated on figure 7.

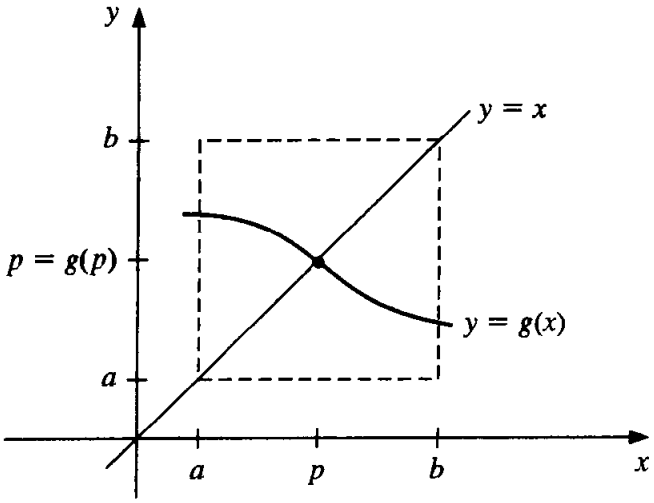


Figure 7. Fixed-point theorem

The main idea of the fixed-point iteration method is following. To approximate the fixed point of a function $g(x)$, we choose an initial approximation p_0 and generate the sequence $\{p_n\}$ by letting $p_n=g(p_{n-1})$, for each $n>1$. If the sequence converges to p and $g(x)$ is continuous, then

$$p = \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g\left(\lim_{n\to\infty} p_{n-1}\right) = g(p).$$

and a solution to $x=g(x)$ is obtained. This technique is called *fixed-point iteration*.

Now we are ready to write the algorithm. Input of the algorithm are initial approximation p_0; tolerance TOL; maximum number of iterations N. Output of the algorithm are solution approximation p or message of failure.

Step 1 Set i = 1.
Step 2 While i < N do Steps 3-6.
Step 3 Set p = g(p_0). //Compute p_i
Step 4 If | p – p_0 | < TOL then
    OUTPUT (x) //The procedure was successful.
    STOP.
Step 5 Set i = i + 1.
Step 6 Set p_0 = p // Update p_0.
Step 7 OUTPUT ('The method failed after N iterations, N =', N0);
    //The procedure was unsuccessful.
    STOP.

Process of finding a fixed point is illustrated on following figure.

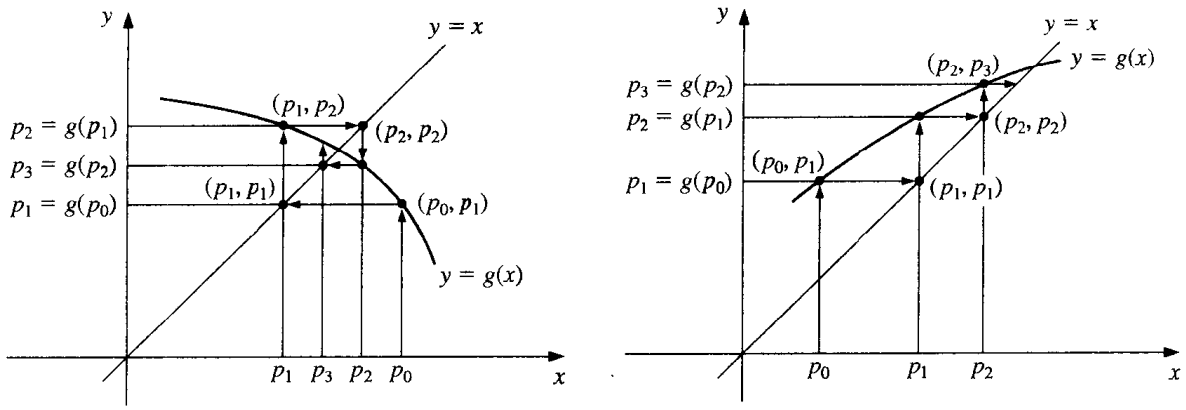


Figure 8. Fixed-point iterations

2.3.1. Error analysis

Theorem. If $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$, and, in addition, $g'(x)$ exists on (a,b) and a constant $0<k<1$ exists with $|g'(x)| \le k$, for all $x \in (a,b)$, then, for any number $p_0 \in [a,b]$, the sequence defined by

$$p_n = g(p_{n-1}), \quad n \ge 1$$

converges to the unique fixed point $p \in [a,b]$.

Corollary. If $g(x)$ satisfies the hypotheses of this theorem, then bounds for the error involved in using p_n to approximate p are given by

$$|p_n - p| \le k^n \max\{p_0 - a, b - p_0\} \quad \text{and} \quad |p_n - p| \le |p_1 - p_0| k^n/(1-k).$$

Proof (theorem). Theorem implies that a unique fixed point $p_0 \in [a,b]$ exists. Using the fact that $|g'(x)| \le k$ and the Mean Value Theorem, we have, for each n,

$$|p_n - p| = |g(p_{n-1}) - g(p)| = |g'(\xi_n)||p_{n-1} - p| \le k|p_{n-1} - p|.$$

where $\xi_n \in (a,b)$. Applying this inequality inductively gives

$$|p_n - p| \le k|p_{n-1} - p| \le k^2|p_{n-2} - p| \le \ldots \le k^n|p_0 - p|.$$

Since $0<k<1$, we have $\lim_{n\to\infty} k^n = 0$ and

$$\lim_{n\to\infty} |p_n - p| \le \lim_{n\to\infty} k^n |p_0 - p| = 0.$$

Proof (corollary). Since $p_0 \in [a,b]$, the first bound follows from inequality

$$|p_n - p| \le k^n |p_0 - p| \le k^n \max\{p_0 - a, b - p_0\}.$$

For $n>1$, the procedure used in the proof of theorem implies that

$$|p_{n+1} - p_n| = |g(p_n) - g(p_{n-1})| \le k|p_n - p_{n-1}| \le \ldots \le k^n |p_1 - p_0|.$$

Thus, for $m>n\ge 1$,

$$\begin{aligned} |p_m - p_n| &= |p_m - p_{m-1} + p_{m-1} - \ldots + p_{n+1} - p_n| \le \\ &\le |p_m - p_{m-1}| + |p_{m-1} - p_{m-2}| + \ldots + |p_{n+1} - p_n| \le \\ &\le k^{m-1}|p_1 - p_0| + k^{m-2}|p_1 - p_0| + \ldots + k^n |p_1 - p_0| = \\ &= k^n |p_1 - p_0|(1 + k + k^2 + \ldots + k^{m-n-1}). \end{aligned}$$

By theorem, $\lim_{m\to\infty} p_m = p$, so

$$|p - p_n| = \lim_{m\to\infty} |p_m - p_n| \le \lim_{m\to\infty} k^n |p_1 - p_0| \sum_{i=0}^{m-n-1} k^i \le k^n |p_1 - p_0| \sum_{i=0}^{\infty} k^i$$

But $\sum_{i=0}^{\infty} k^i$ is a geometric series with ratio k and $0<k<1$. This sequence converges to $\frac{1}{1-k}$, which gives the second bound

$$|p - p_n| \le \frac{k^n}{1-k} |p_1 - p_0|.$$

Both inequalities in the corollary relate the rate at which $\{p_n\}$ converges to the bound k on the first derivative. The rate of convergence depends on the factor k^n. The smaller the value of k, the faster the convergence, which may be very slow if k is close to 1.

Example 1. For $g(x)=x-x^3-4x^2+10$, we have $g(1) = 6$ and $g(2)=-12$, so $g(x)$ does not map [1,2] into itself. Moreover, $g'(x)=1-3x^2-8x$, so $|g'(x)|>1$ for all x in [1,2]. Although Theorem does not guarantee that the method must fail for this choice of g, there is no reason to expect convergence.

Example 2. For $g(x)=(10/(4+x))^{0.5}$, we have for all x in [1,2]

$$|g'(x)| = \left|\frac{-5}{\sqrt{10}(4+x)^{3/2}}\right| \le \frac{5}{\sqrt{10}(5)^{3/2}} < 0.15$$

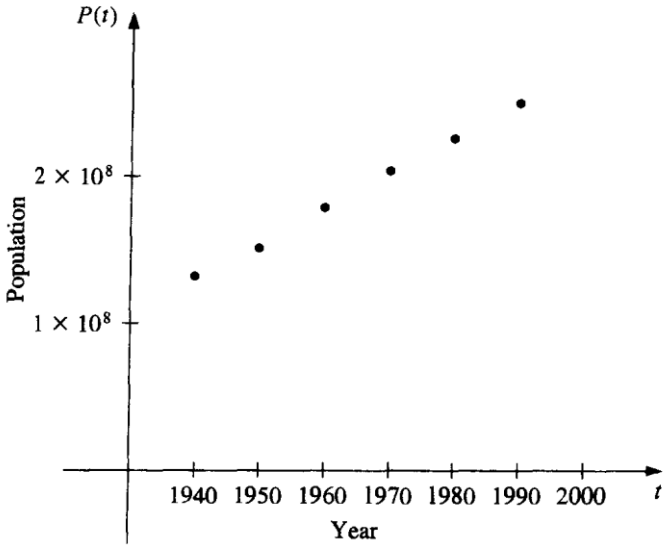
The bound on the magnitude of $g'(x)$ is small, which explains the rapid convergence.

3. Interpolation and polynomial approximation

3.1. Problem statement

A census of the population of the United States is taken every 10 years. The following table lists the population, in thousands of people, from 1940 to 1990.

Year	Population (in ths)
1940	132,165
1950	151,326
1960	179,323
1970	203,302
1980	226,542
1990	249,633



We might ask whether they could be used to provide a reasonable estimate of the population, say, in 1965 or in 1995. Predictions of this type can be obtained by using a function that fits the given data. This process is called *interpolation* and is the subject of this chapter.

One of the most useful and well-known classes of functions mapping the set of real numbers into itself is the class of *algebraic polynomials*, the set of functions of the form

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0$$

where n is a nonnegative integer and $a_0, a_1, \ldots, a_n$ are real constants. One reason for their importance is that they uniformly approximate continuous functions. Given any function, defined and continuous on a closed and bounded interval, there exists a polynomial that is as "close" to the given function as desired.

Theorem. (Weierstrass Approximation Theorem) without proof.

Suppose that $f(x)$ is defined and continuous on $[a,b]$. For each $\varepsilon>0$, there exists a polynomial $P(x)$, with the property that

$$|f(x) - P(x)| < \varepsilon, \qquad \text{for all } x \text{ in } [a,b].$$

Weierstrass Approximation Theorem is illustrated on the following figure.

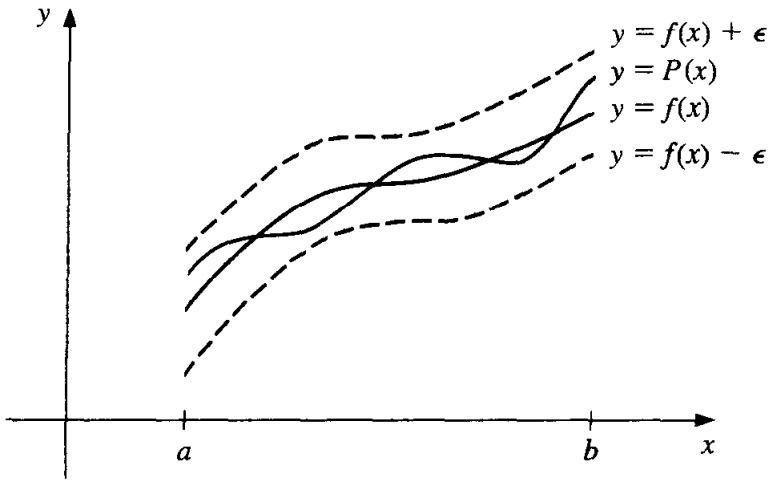


Figure 9. Weierstrass Approximation Theorem

3.2. Linear interpolation

The problem of determining a polynomial of degree one that passes through the distinct points (x_0,y_0) and (x_1,y_1) is the same as approximating a function f for which $f(x_0)=y_0$ and $f(x_1)=y_1$ by means of a first-degree polynomial interpolating, or agreeing with, the values of f at the given points. We first define the functions

$$L_0(x) = \frac{x-x_1}{x_0-x_1} \text{ and } L_1(x) = \frac{x-x_0}{x_1-x_0}.$$

and then define

$$P(x) = L_0(x) f(x_0) + L_1(x) f(x_1)$$

Since $L_0(x_0)=1$, $L_0(x_1)=0$, $L_1(x_0)=0$, $L_1(x_1)=1$, we have

$$P(x_0)=1\ f(x_0)+0\ f(x_1)=f(x_0)=y_0$$
$$P(x_1)=0\ f(x_0)+1\ f(x_1)=f(x_1)=y_1$$

So P is the unique linear function passing through (x_0,y_0) and (x_1,y_1). This fact is illustrated on the following figure.

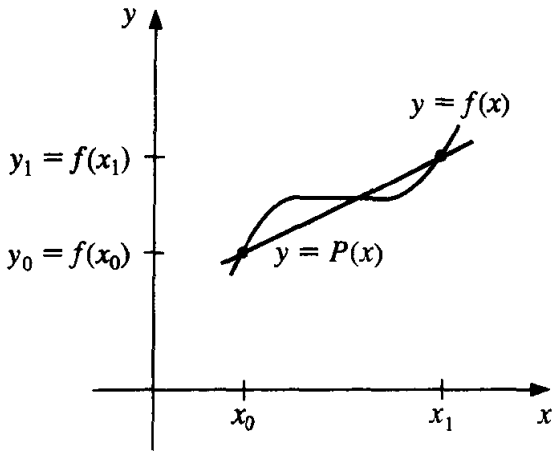


Figure 10. Linear interpolation

3.3. General interpolation problem

To generalize the concept of linear interpolation, consider the construction of a polynomial of degree at most n that passes through the n+1 points

$$(x_0,f(x_0)),\ (x_1,f(x_1)),\ldots,\ (x_n,f(x_n)).$$

This problem is illustrated on figure 11.

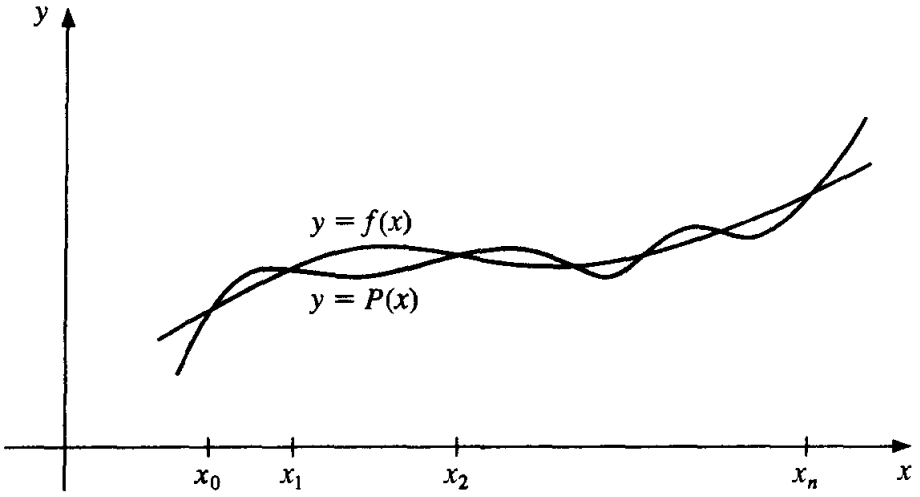


Figure 11. Polynomial interpolation

In this chapter, we solve the following problem: we are given a table of n+1 data points (x_i,y_i):

x_i	x_0	x_1	...	x_n
y_i	y_0	y_1	...	y_n

and we seek a polynomial $p(x)$ of lowest possible degree for which

$$p(x_i)=y_i,\ \text{where } 0\le i\le n.$$

Such a polynomial is said to *interpolate* the data, or $p(x)$ is called *interpolating polynomial*.

Theorem 1. If x_0, $x_1,\ldots,$ x_n are distinct real numbers, then for arbitrary values $y_0,y_1,\ldots,y_n$ there is a unique polynomial $p_n(x)$ of degree at most n such that

$$p(x_i)=y_i, \text{ where } 0\le i\le n.$$

Proof. (Unicity) Suppose there were two such polynomials, $p_n(x)$ and $q_n(x)$. Then the polynomial $p_n(x)-q_n(x)$ would have the property that $p_n(x_i)-q_n(x_i)=0$ for $0\le i\le n$. Since the degree of $p_n(x)-q_n(x)$ can be at most n, this polynomial can have at most n zeros if it is not the zero polynomial. Since the x_i, are distinct, $p_n(x)-q_n(x)$ has $n+1$ zeros; it must therefore be zero. Hence, $p_n(x) \equiv q_n(x)$.

(Existence) For $n=0$, the existence is obvious since a constant function p_0 (polynomial of degree 0) can be chosen so that $p_0(x_0)=y_0$. Now suppose that we have obtained a polynomial p_{k-1} of degree $k-1$ with $p_{k-1}(x_i)=y_i$ for $0\le i\le k-1$.

We try to construct p_k in the form

$$p_k(x)=p_{k-1}(x)+c(x-x_0)\,(x-x_1)\ldots(x-x_{k-1})$$

Note that this is unquestionably a polynomial of degree at most k. Furthermore, p_k interpolates the data that p_{k-1} interpolates, because

$$p_k(x_i)=p_{k-1}(x_i)=y_i, \quad 0\le i\le k-1.$$

Now we determine the unknown coefficient c from the condition $p_k(x_k)=y_k$. This leads to the equation

$$p_{k-1}(x_k)+c(x_k-x_0)\,(x_k-x_1)\ldots(x_k-x_{k-1})=y_k.$$

This equation can certainly be solved for c because the factors multiplying c are not zero. Theorem is proofed.

Theorem 2. If x_0, $x_1,\ldots,$ x_n are distinct real numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$P(x_k)=f(x_k), \text{ where } 0\le k\le n.$$

This polynomial is given by

$$P(x) = f(x_0)L_{n,0}(x) + \cdots f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x)$$

where

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}$$

or

$$L_{n,k}(x) = \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)}$$

We will write $L_{n,k}(x)$ simply as $L_k(x)$ when there is no confusion as to its degree.

Theorem 3. (without proof).

Suppose that $x_0, x_1, \ldots, x_n$ are distinct numbers in the interval $[a,b]$ and $f \in C^{n+1}[a,b]$. Then, for each $x \in [a,b]$, a number $\xi(x) \in (a,b)$ exists with

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \ldots (x - x_n)$$

where $P(x)$ is the interpolating polynomial.

Corollary. Error of interpolation is

$$E = f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \ldots (x - x_n)$$

4. Numerical differentiation

4.1. Problem statement

If the values of a function $f(x)$ are given at a few points, say $x_0, x_1, \ldots, x_n$, can that information be used to estimate a derivative $f'(c)$? The answer is a qualified *Yes*.

Let us begin by observing that from the values $f(x_i)$ alone it is impossible to infer very much about $f(x)$ unless we are informed also that $f(x)$ belongs to some relatively small family of functions. Thus, if $f(x)$ is allowed to range over the family of all continuous real-valued functions, the values $f(x_i)$ are almost useless.

Figure 12 illustrates several continuous functions taking the same values at six points.

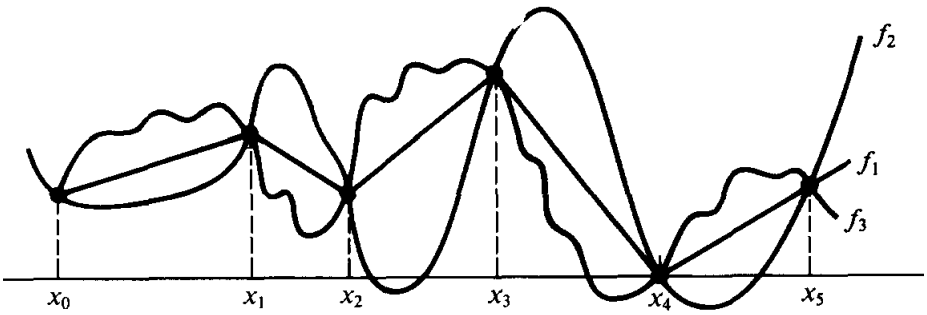


Figure 12. Several continuous functions

If we know that $f(x)$ is a polynomial of degree at most n, then the values at $n+1$ points completely determine $f(x)$, by the theory of interpolation. In this case, we recover $P(x)$ precisely, and can then compute $P'(c)$. And then we can use $P'(c)$ as estimation of $f'(c)$.

4.2. Two-point formulas for first derivative

First we consider a formula for numerical differentiation that emerges directly from the definition of $f'(x)$:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

This formula gives an obvious way to generate an approximation to $f'(x)$: simply compute

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \qquad (*)$$

For a linear function, $f(x)=ax+b$, the approximate formula is exact; that is, it yields the correct value of $f'(x)$: for any nonzero value of h. Formula (*) is known as the *forward-difference formula* if $h>0$ and the *backward-difference formula* if $h<0$.

Figure 13 illustrates the two-point formula graphically.

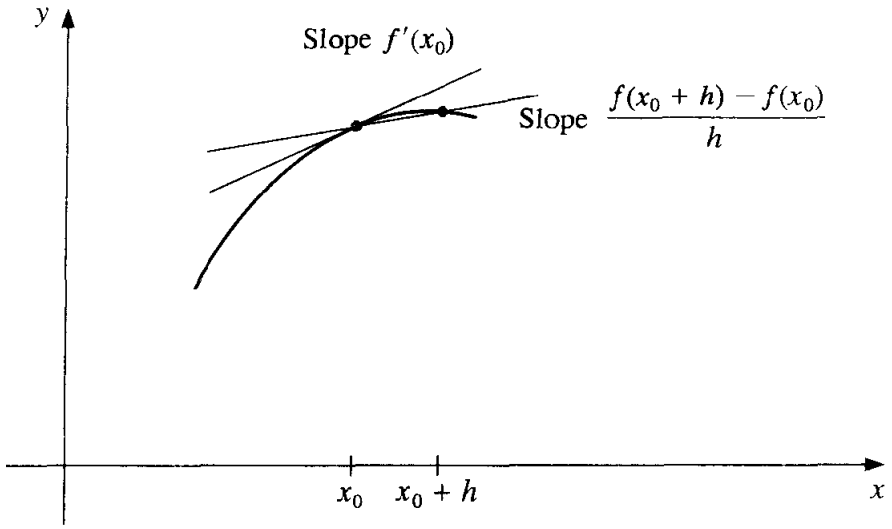


Figure 13. Two point formula

4.2.1. Error analysis

The starting point is Taylor's Theorem in this form

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\xi).$$

Here ξ is a point in the open interval between x and $x+h$. For the validity of equation, f and f' should be continuous on the closed interval between x and $x+h$, and f'' should exist on the corresponding open interval. A rearrangement of equation yields

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(\xi)$$

Now *error term* is available along with the basic numerical formula. Notice that the error term in equation has two parts: a power of h and a factor involving some higher-order derivative of f. The h-term in the error makes the entire expression converge to zero as h approaches zero.

4.3. Three-point formula for the first derivative

The precision of such numerical differentiation formulae is often judged simply by the power of h present in the error term. The higher the power of h the better, for h is always a small number. In this assessment, formula (*) fares poorly, as the error is $O(h)$. A superior formula is

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

This is derived from two cases of Taylor's Theorem, namely

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(\xi_1),$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(\xi_2).$$

On subtracting one of these from the other, we obtain

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{12}\left(f'''(\xi_1) + f'''(\xi_2)\right)$$

Since f'' is continuous some point ξ exists in [x–h, x+h] such, that

$$f'''(\xi) = 0.5\left(f'''(\xi_1) + f'''(\xi_2)\right)$$

When this expression is substituted in equation previous, the result is

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(\xi).$$

Three-point formula is illustrated on the following figure.

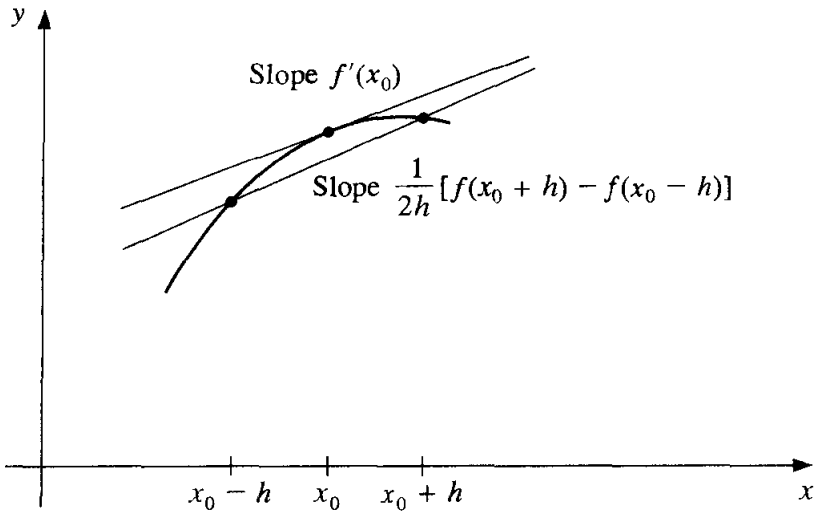


Figure 14. Three-point formula

4.4. Three-point formula for the second derivative

An important formula for second derivatives is obtained at the same way from Teylor's theorem

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(x) + \frac{h^4}{24} f^{(4)}(\xi_1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(x) + \frac{h^4}{24} f^{(4)}(\xi_2)$$

On adding one of these to the other, we obtain

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{h^2}{12} f^{(4)}(\xi)$$

where $\xi \in [x-h, x+h]$ and $f^{(4)}(\xi) = 0.5\left(f^{(4)}(\xi_1) + f^{(4)}(\xi_2)\right)$.

Finally, we have

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

4.5. Differentiation via interpolation

A general approach to numerical differentiation and integration can be based on polynomial interpolation. Suppose that we have $n+1$ values of a function f at points $x_0, x_1, \ldots, x_n$. A polynomial that interpolates f at the nodes x_i can be written in the Lagrange form as

$$f(x) = \sum_{i=0}^{n} L_i(x) f(x_i) + \frac{f^{(n+1)}(\xi)}{(n+1)!} w(x)$$

where $w(x) = \prod_{i=0}^{n} (x - x_i)$.

Taking derivative in this equation we have

$$f'(x) = \sum_{i=0}^{n} L_i'(x) f(x_i) + \frac{f^{(n+1)}(\xi)}{(n+1)!} w'(x) + \frac{w(x)}{(n+1)!} \frac{d}{dx} f^{(n+1)}(\xi)$$

If x is one of the nodes, say $x=x_\alpha$, then the preceding equation simplifies, since $w(x_\alpha)=0$, and the result is

$$f'(x) = \sum_{i=0}^{n} L_i'(x_\alpha) f(x_i) + \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{\substack{i=0 \\ i \neq \alpha}}^{n} (x_\alpha - x_i)$$

and derivative can be calculated as

$$f'(x) \approx \sum_{i=0}^{n} L_i'(x_\alpha) f(x_i) \qquad \text{(X)}$$

$f'''(x) \approx \sum_{i=0}^{n} L_i'''(x_\alpha) f(x_i)$? $f''(x) \approx \sum_{i=0}^{n} L_i''(x_\alpha) f(x_i)$, and so on.

4.6. Exercises

1. Give the explicit form of equation (X) when n=2 and α=0.
2. Give the explicit form of equation (X) when n=2 and α=2.
3. Estimate error of approximation using Teylor's theorem.

5. Numerical integration

5.1. Problem statement

A sheet of corrugated roofing is constructed by pressing a flat sheet of aluminum into one whose cross section has the form of a sine wave.

Figure 15. A sheet of corrugated roofing

A corrugated sheet 4 ft. long is needed, the height of each wave is 1in. from the center line, and each wave has a period of approximately 2π in. The problem of finding the length of the initial flat sheet is one of determining the length of the curve given by $f(x)=\sin(x)$ from $x=0$ to x=48. From mathematical analysis a we know that this length is

$$L=\int_0^{48}\sqrt{1+\left(f'(x)\right)^2}\,dx=\int_0^{48}\sqrt{1+\left(\cos x\right)^2}\,dx,$$

so the problem reduces to evaluating this integral. Although the sine function is one of the most common mathematical functions, the calculation of its length involves an elliptic integral of the second kind, which cannot be evaluated by ordinary methods. Methods of approximation the solution to problems of this type are considered in this chapter.

Numerical integration is the process of producing a numerical value for the integration of a function over a set. For example, following integration problems are not amenable to the techniques learned in elementary calculus.

$$\int_0^1\int_0^1 \sin(xy\exp(x))dxdy$$

$$\int_0^{\pi}\cos(3\cos x)dx$$

Those techniques depends on *antidifferentiation*. Thus, to find the value of the integral, we first must produce a function F with the property that $F'=f$. Then, we have

$$\int_a^b f(x)dx=F(a)-F(b).$$

There are many elementary functions that do not have simple antiderivatives. A good example is $f(x)=e^{x^2}$.

5.2. Integration via interpolation

One powerful stratagem for computing the integral numerically is to replace $f(x)$ by another function $g(x)$ that approximates $f(x)$ well and is easily integrated. Then, we simply say to ourselves that from $f(x) \approx g(x)$ it follows that

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

Polynomials are good candidates for the function g, and indeed, $g(x)$ can be a polynomial that interpolates to $f(x)$ at a certain set of nodes.

Suppose that we want to evaluate the integral in $[a,b]$. We can select nodes in $[a,b]$ and set up a Lagrange polynomial

$$P_n(x) = \sum_{i=0}^{n} \left[L_i(x) f(x_i) \right], \text{ where } L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

Then, as mentioned previously, we simply write

$$\int_a^b f(x)dx \approx \int_a^b P_n(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_n(x)dx$$

In this way, we obtain a formula that can be used on any f.

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i), \text{ where } A_i = \int_a^b L_i(x)dx.$$

This formula is called a *Newton-Cotes formula*, if the nodes are equally spaced.

5.3. Trapezoid rule

The simplest case results if $n=1$ and the nodes are $x_0=a$, $x_1=b$. In this case

$$L_0(x) = \frac{b-x}{b-a}, \; L_1(x) = \frac{x-a}{b-a}.$$

Consequently,

$$A_0 = \int_a^b L_0(x)dx = 0.5(b-a) = \int_a^b L_1(x)dx = A_1$$

The corresponding quadrature formula is

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \left[f(a) + f(b) \right].$$

This is called the trapezoid rule because when $f(x)$ is a function with positive values, integral is approximated by the area in a trapezoid, as shown in figure.

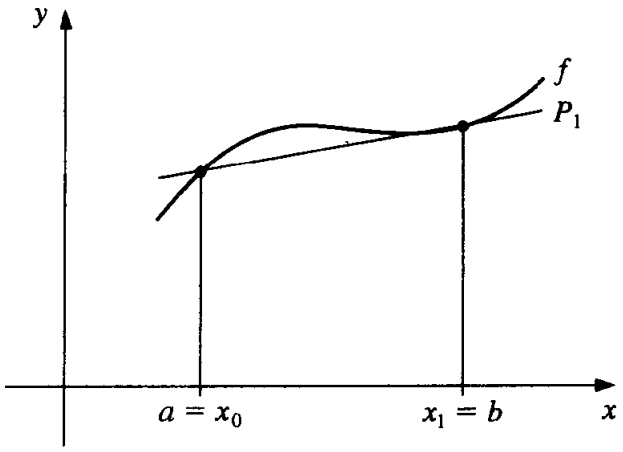


Figure 16. Trapezoid rule

Error analysis of trapezoid rule is presented below. As far as

$$f(x) = \sum_{i=0}^{n} L_i(x) f(x_i) + \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

the error of integration is

$$E(f) = \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)dx$$

In the case of trapezoid formula n=1, x_0=a, x_1=b, and error is

$$E(f) = \frac{f^{(2)}(\xi)}{2!} \int_a^b (x - x_0)(x - x_1)dx = \frac{f''(\xi)}{2} \left[\frac{x^3}{3} - \frac{x_0 + x_1}{2} x^2 + x_0 x_1 x \right]_{x_0}^{x_1} = -\frac{h^3}{12} f''(\xi) .$$

5.4. Simpson's rule

Simpson's rule results from integrating over [a,b] the second Lagrange polynomial with nodes x_0=a, x_2=b, and x_1=a+h, where h=(b–a)/2. In this case

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, \; L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, \; L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

and

$$A_0 = \int_a^b L_0(x)dx = \frac{h}{3} = \int_a^b L_1(x)dx = A_2, \; A_1 = \int_a^b L_1(x)dx = \frac{4h}{3}.$$

Consequently,

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)].$$

Error of this formula is

$$E(f) = -\frac{h^5}{90} f^{(4)}(\xi), \text{ where } \xi \in [a,b].$$

Simpson's rule results from integrating over [a,b] the second Lagrange polynomial.

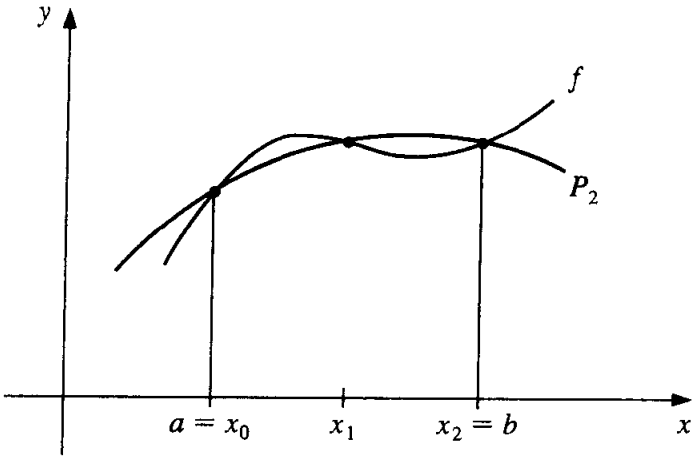
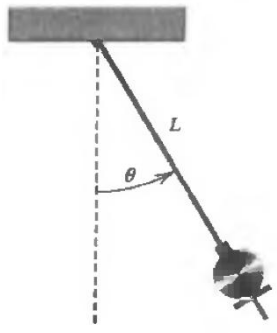


Figure 17. Simpson's rule

6. Initial-value problems for ODE

6.1. Problem statement



The motion of a swinging pendulum under certain simplifying assumptions is described by the second-order differential equation

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin(\theta) = 0,$$

where L is the length of the pendulum, g is the gravitational constant of the earth, and θ is the angle the pendulum makes with the vertical. If, in addition, we specify the position of the pendulum when the motion begins, $\theta(t_0)=\theta_0$, and its velocity at that point, $\theta'(t_0) = \theta_0'$, we have what is called an *initial-value problem*.

For small values of θ, the approximation $\theta \approx \sin\theta$ can be used to simplify this problem to the linear initial-value problem

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0, \theta(t_0)=\theta_0, \theta'(t_0) = \theta_0'.$$

This problem can be solved by a standard differential-equation technique. For larger values of θ, approximation methods must be used.

Any textbook on ordinary differential equations details a number of methods for explicitly finding solutions to first-order initial-value problems. In practice, however, few of the problems originating from the study of physical phenomena can be solved exactly.

The first part of this chapter is concerned with approximating the solution $y(x)$ to a problem of the form

$$\frac{dy}{dx} = f(x, y) \quad \text{for } a \le x \le b$$

subject to an initial condition $y(a)=y_0$.

Later in the chapter we deal with the extension of these methods to a system of first-order differential equations. We also examine the relationship of a system of this type to the general nth-order initial-value problem.

6.2. Taylor-series method

To illustrate the method we take a concrete example

$$y' = \cos x - \sin y + x^2, \qquad y(-1) = 3.$$

At the heart of the procedure is the Taylor series for x, which we write as

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2} y''(x) + \frac{h^3}{6} y'''(x) + O(h^4)$$

The derivatives appearing here can be obtained from the differential equation. They are

$$y'' = -\sin x - y' \cos y + 2x,$$

$$y''' = -\cos x - y'' \cos x + (y')^2 \sin x + 2.$$

We decide to use only terms up to and including h^3 in the formula. The terms that we have not included start with a term in h^4, and they constitute collectively the *truncation error* inherent in our procedure. The resulting numerical method is said to be of order 3. (The *order* of the Taylor-series method is n if terms up to and including h^n are used.)

We could perform various substitutions to obtain formulae for x'', x''',... containing no derivatives of x on the right-hand side.

6.2.1. Error analysis

At each step, the *local* truncation error is $O(h^4)$ since we have not included terms involving h^4, h^5,... from the Taylor series. Thus, as h $\to$0, the behavior of the local errors should be similar to Ch^4. Unfortunately, we do not know C. But h^4 is 10^{-8} since h=10^{-2}. So with good luck, the error in each step should be roughly of the magnitude 10^{-8}. After several hundred steps, these small errors could accumulate and spoil the numerical solution. At each step (except the first), the estimate y_k of $y(x_k)$ already contains errors, and further computations continue to add to these errors.

The *local truncation error* is the error made in one step when we replace an infinite process by a finite one. In the Taylor-series method, we replace the infinite Taylor series for $y(x+h)$ by a partial sum.

The accumulation of all these many local truncation errors gives rise to the *global truncation error*.

If the local truncation errors are $O(h^{n+1})$, then the global truncation error must be $O(h^n)$ because the number of steps necessary to reach an arbitrary point x, having started at x_0, is $(x–x_0)/h$.

6.3. Euler's method

The Taylor-series method with n=1 is called *Euler's method*. It looks like this

$$y(x_i + h) = y(x_i) + hf(x_i, y_i).$$

This formula has the obvious advantage of not requiring any differentiation of $f(x,y)$.

Local truncation error of Euler's method is $O(h^2)$, then the global truncation error must be $O(h)$. Euler's method is a first-order method.

To interpret Euler's method geometrically, note that when y_i is a close approximation to $y(x_i)$, the assumption that the problem is well-posed implies that

$$f(x_i,y_i) \approx y'(x_i) = f(x_i, y(x_i))$$

One step in Euler's method is shown in left figure, and a series of steps appears in right figure.

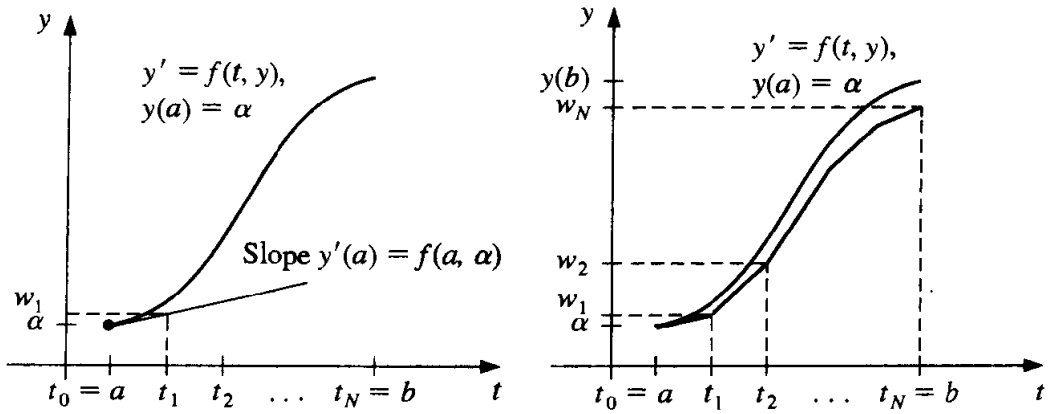


Figure 18. Euler's method

6.4. Second-order Runge-Kutta methods

Let's begin with the Taylor series for $y(x+h)$

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2} y''(x) + O(h^3)$$

From the differential equation, we have

$$y'(x) = f(x,y)$$

$$y''(x) = f'_x + f'_y y' = f'_x + f'_y f$$

Here subscripts denote partial derivatives. The first three terms in Taylor series can be written now in the form

$$y(x+h) = y(x) + hf + \frac{h^2}{2}(f'_x + f'_y f) + O(h^3) = \quad (*)$$

$$= y(x) + \frac{1}{2}hf + \frac{1}{2}h(f + hf'_x + hf'_y f) + O(h^3)$$

We are able to eliminate the partial derivatives with the aid of the first few terms in the Taylor series in two variables

$$f(x+h, y+hf) = f(x,y) + hf'_x + hf'_y y' + O(h^2) =$$
$$= f(x,y) + hf'_x + hf'_y f + O(h^2)$$

Taylor series can be rewritten as

$$y(x+h) = y(x) + \frac{1}{2}hf + \frac{1}{2}hf(x+h, y+hf) + O(h^3)$$

Hence, the formula for advancing the solution is

$$y(x_i + h) = y(x_i) + \frac{1}{2}hf(x_i, y_i) + \frac{1}{2}hf(x_i + h, y_i + hf(x_i, y_i))$$

This formula can be used repeatedly to advance the solution one step at a time. It is called a second-order Runge-Kutta method. It is also known as *Heun's method*.

In general, second-order Runge-Kutta formulas are of the form

$$y(x+h) = y(x) + w_1 hf + w_2 hf(x + \alpha h, y + \beta hf) + O(h^3)$$

where w_1, w_2, α, β are parameters at our disposal. This equation can be rewritten with the aid of the Taylor series in two variables as

$$y(x+h) = y(x) + w_1 hf + w_2 h\left[f + \alpha h f_x + \beta h f_y f\right] + O(h^3) \quad (**)$$

Comparing (*) with (**), we see that we should impose these conditions

$$\begin{cases} w_1 + w_2 = 1 \\ w_2 \alpha = 0.5 \\ w_2 \beta = 0.5 \end{cases}$$

One solution is w_1=w_2=0.5, α=β=1, which is the one corresponding to Heun's method.

The system of equations has solutions other than this one, such as the one obtained by letting w_1=0, w_2=1, α=β=0.5. The resulting formula is called the *modified Euler method*

$$y(x_i + h) = y(x_i) + hf(x_i + 0.5h, y_i + 0.5hf(x_i, y_i)).$$

6.5. Fourth-order Runge-Kutta method

The higher-order Runge-Kutta formulas are very tedious to derive, and we shall not do so. The formulae are rather elegant, however, and are easily programmed once they have been derived. Here are the formulae for the classical *fourth-order Runge-Kutta method*

$$y(x_i + h) = y(x_i) + \frac{1}{6}(F_1 + 2F_2 + 2F_3 + F_4)$$

where

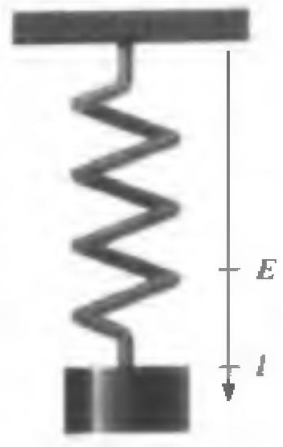
$$F_1 = hf(x_i, y_i)$$
$$F_2 = hf(x_i + 0.5h, y_i + 0.5F_1)$$
$$F_3 = hf(x_i + 0.5h, y_i + 0.5F_2)$$
$$F_4 = hf(x_i + h, y_i + F_3)$$

This is called a fourth-order method because it reproduces the terms in the Taylor series up to and including the one involving h^4. The local error is therefore $O(h^5)$, global error is $O(h^4)$.

7. Curve fitting

7.1. Problem statement

Hooke's law states that when a force is applied to a spring, the length of the spring is a linear function of that force. We can write the linear function as $F(l)=k(l-E)$, where $F(l)$ is the force required to stretch the spring l units, the constant E is the length of the spring with no force applied, and the constant k is the spring constant.



Suppose we want to determine the spring constant for a spring that has initial length 5.3. We apply forces of 2, 4, and 6 to and find that its length increases to 7.0, 9.4, and 12.3. A quick examination shows that the points (0,5.3), (2,7.0), (4,9.4), and (6,12.3) do not quite lie in a straight line. Although we could simply use one random pair of these data points to approximate the spring constant, it would seem more reasonable to find the line that *best approximates* all the data points to determine the constant. This type of approximation will be considered in this chapter.

The study of approximation theory involves two general types of problems. One problem arises when a function is given explicitly, but we wish to find a "simpler" type of function, such as a polynomial, that can be used to determine approximate values of the given function. The other problem in approximation theory is concerned with fitting functions to given data and finding the "best" function in a certain class to represent the data.

Both problems have been considered in previous chapters. The Taylor polynomial of degree n about the number x_0 is an excellent approximation to an (n+l)-times differentiable function $f(x)$ in a small neighborhood of x_0. The Lagrange interpolating polynomial of degree n is used both as approximating polynomial and as polynomial to fit certain data. Now we consider the problem of finding the "best" function in a certain class to represent the certain data.

Consider the problem of estimating the values of a function at nontabulated points, given the experimental data in following table. A graph of the values in table is shown on a figure.

x_i	y_i
1	1,3
2	3,5
3	4,2
4	5,0

5	7,0
6	8,8
7	10,1
8	12,5
9	13,0
10	15,6

From this graph, it appears that the actual relationship between *x* and *y* is linear. The likely reason that no line precisely fits the data is because of errors in the data. So it is unreasonable to require that the approximating function agree exactly with the data. In fact, such a function would introduce oscillations that were not originally present. For example, we can approximate this data with 9-th degree interpolating polynomial (it is shown on the following figure).

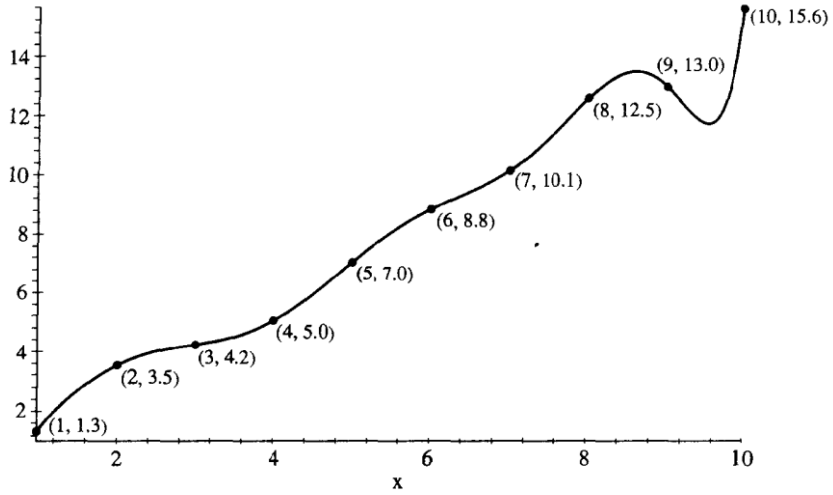


Figure 19. Interpolating polynomial

This polynomial is clearly a poor predictor of information between a number of the data points. A better approach would be to find the "best" (in some sense) approximating line, even if it does not agree precisely with the data at any point.

Minimax problem. Let $a_1x_i+a_0$ denote the *i*th value on the approximating line and y_i be the *i*th given *y*-value. The problem of finding the equation of the best linear approximation in the absolute sense requires that values of a_0 and a_1 be found to minimize

$$E_\infty(a_0, a_1) = \max_{1\le i\le 10}\{|y_i - (a_1x_i + a_0)|\}.$$

This is commonly called a *minimax problem* and cannot be handled by elementary techniques.

Absolute deviation. Another approach to determining the best linear approximation involves finding values of a_0 and a_1 to minimize

$$E_1(a_0, a_1) = \sum_{i=1}^{10} |y_i - (a_1x_i + a_0)|.$$

This quantity is called the *absolute deviation*. To minimize a function of two variables, we need to set its partial derivatives to zero and simultaneously solve the resulting equations. In the case of the absolute deviation, we need to find a_0 and a_1 with

$$\frac{\partial}{\partial a_0}\sum_{i=1}^{10}|y_i-(a_1x_i+a_0)|=0$$

$$\frac{\partial}{\partial a_1}\sum_{i=1}^{10}|y_i-(a_1x_i+a_0)|=0$$

The difficulty is that the absolute-value function is not differentiable at zero, and we may not be able to find solutions to this pair of equations.

7.2. Least squares line

The *least squares* approach to this problem involves determining the best approximating line when the error involved is the sum of the squares of the differences between the *y*-values on the approximating line and the given *y*-values. Hence, constants a_0 and a_1 must be found that minimize the least squares error

$$E_2(a_0,a_1)=\sum_{i=1}^{10}[y_i-(a_1x_i+a_0)]^2.$$

The least squares method is the most convenient procedure for determining best linear approximations. The minimax approach generally assigns too much weight to a bit of data that is badly in error, whereas the absolute deviation method does not give sufficient weight to a point that is considerably out of line with the approximation. The least squares approach puts substantially more weight on a point that is out of line with the rest of the data but will not allow that point to completely dominate the approximation.

The general problem of fitting the best least squares line to a collection of data $\{(x_i,y_i)\}$, $i=1,\ldots,m$, involves minimizing the total error

$$E\equiv E_2(a_0,a_1)=\sum_{i=1}^{m}[y_i-(a_1x_i+a_0)]^2.$$

with respect to the parameters a_0 and a_1. For a minimum to occur, we need

$$0=\frac{\partial}{\partial a_0}\sum_{i=1}^{10}[y_i-(a_1x_i+a_0)]^2=2\sum_{i=1}^{10}[y_i-(a_1x_i+a_0)](-1)$$

$$0=\frac{\partial}{\partial a_1}\sum_{i=1}^{10}[y_i-(a_1x_i+a_0)]^2=2\sum_{i=1}^{10}[y_i-(a_1x_i+a_0)](-x_i)$$

These equations simplify to the *normal equations*

$$\begin{cases} a_0 m + a_1 \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} y_i \\ a_0 \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2 = \sum_{i=1}^{m} x_i y_i \end{cases}$$

The solution of this normal equations system is obtained by Kramer's rule or Gauss method.

Example. Let's calculate the least squares line for the data from the previous example.

	x_i	y_i	x_i^2	$x_i y_i$
	1,00	1,30	1,00	1,30
	2,00	3,50	4,00	7,00
	3,00	4,20	9,00	12,60
	4,00	5,00	16,00	20,00
	5,00	7,00	25,00	35,00
	6,00	8,80	36,00	52,80
	7,00	10,10	49,00	70,70
	8,00	12,50	64,00	100,00
	9,00	13,00	81,00	117,00
	10,00	15,60	100,00	156,00
Sum	55,00	81,00	385,00	572,40

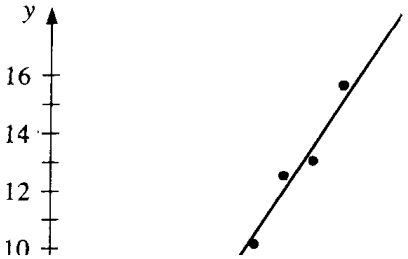
The normal equations imply that

$$a_0 = \frac{385 \cdot 81 - 55 \cdot 572.4}{10 \cdot 385 - 55^2} = -0.36$$

$$a_1 = \frac{10 \cdot 572.4 - 55 \cdot 81}{10 \cdot 385 - 55^2} = 1.538$$

So $P(x)=1.538\,x - 0.36$

The graph of this line and the data points are shown in figure. The approximate values given by the least squares technique at the data points are in following table.



x_i	y_i	$P(x_i)$
1,00	1,30	1,18
2,00	3,50	2,72
3,00	4,20	4,25
4,00	5,00	5,79

5,00	7,00	7,33
6,00	8,80	8,87
7,00	10,10	10,41
8,00	12,50	11,94
9,00	13,00	13,48
10,00	15,60	15,02

7.3. Least squares polynomial

The general problem of approximating a set of data $\{(x_i,y_i)\}$, i=1,…,m, with an algebraic polynomial

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0$$

of degree n<m–1, using the least squares procedure is handled in a similar manner. We choose the constants a_0, a_1,... , a_n to minimize the least squares error

$$E_2 = \sum_{i=1}^{m} [y_i - P_n(x_i)]^2 = \sum_{i=1}^{m} [y_i]^2 - 2\sum_{i=1}^{m} y_i P_n(x_i) + \sum_{i=1}^{m} [P_n(x_i)]^2$$

As in the linear case, for E to be minimized it is necessary that $\partial E/\partial a_j = 0$ for each j=0,…, m. Thus, for each j,

$$0 = \frac{\partial E}{\partial a_j} = -2\sum_{i=1}^{m} y_i x_i^j + 2\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k}.$$

This gives n+1 *normal equations* in the n+1 unknowns a_j

$$\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k} = \sum_{i=1}^{m} y_i x_i^j \text{ for each } j = 0, \ldots, n.$$

These normal equations have a unique solution provided that the x_j are distinct. Error of approximation can be written in the following form

$$E = \sum_{i=1}^{m} \left(y_i - P(x_i)\right)^2 .$$

Home exercises

Finite digit arithmetic. Absolute and relative error.

1. Find intervals containing solutions to the following equations.
   a. $x - 3^{-x} = 0$
   b. $4x^2 - 2^x = 0$
2. Show that the following equations have at least one solution in the given intervals.
   a. $x\cos(x) - 2x^2 + 3x - 1 = 0$, [0.2, 0.3] and [1.2, 1.3]

b. $(x-2)^2 - \ln(x) = 0$, [1,2] and [e,4]

c. $2x\cos(2x) - (x-2)^2 = 0$, [2, 3] and [3,4]

3. Find the largest interval in which p^* must lie to approximate p with relative error at most 10^{-4} for each value of p.

a. $p = \pi$

b. p=e

c. $p=\sqrt{2}$

4. Suppose p^* must approximate p with relative error at most 10^{-3}. Find the largest interval in which p^* must lie for each value of p.

a. p=150

b. p=900

c. p=1500

d. p=90

5. Compute the absolute error and relative error in approximations of p by p^*.

a. $p = \pi$, $p^* = 22/7$ b. $p = \pi$, $p^* = 3.1416$

c. $p = e$, $p^* = 2.718$ d. $p = \sqrt{2}$, $p^* = 1.414$

e. $p = e^{10}$, $p^* = 22000$ f. $p = 10^{\pi}$, $p^* = 1400$

Solution of nonlinear equations. Bisection method.

1. Let $f(x) = (x+2)(x+1)^2x(x-1)^3(x-2)$. To which zero of $f(x)$ does the Bisection method converge when applied on the following intervals?

a. [-1.5,2.5]

b. [-0.5,2.4]

c. [-0.5,3]

d. [-3,-0.5]

2. Let $f(x) = (x+2)(x+1)\,x(x-1)^3(x-2)$. To which zero of $f(x)$ does the Bisection method converge when applied on the following intervals?

a. [-3,2.5]

b. [-2.5,3]

c. [-1.75,1.5]

d. [-1.5,1.75]

3. Use Theorem to find a bound for the number of iterations needed to achieve an approximation with relative error 10^{-3} to the solution of $x^3 + x - 4 = 0$ lying in the interval [1, 4].

4. Use Theorem to find a bound for the number of iterations needed to achieve an approximation with absolute error 10^{-6} to the solution of $x^3 - x - 1 = 0$ lying in the interval [1, 2].

5. Write a program that implements the Bisection method and find solutions accurate to within 10^{-10} for the following problems:

a. $x - 2^{-x} = 0$ for $0 \le x \le 1$

b. $e^x - x^2 + 3x - 2 = 0$ for $0 \le x \le 1$

c. $2x\cos(2x) - (x+1)^2 = 0$ for $-3 \le x \le -2$ and $-1 \le x \le 0$

d. $x\cos(x) - 2x^2 + 3x - 1 = 0$ for $0.2 \le x \le 0.3$ and $1.2 \le x \le 1.3$

Solution of nonlinear equations. Newton's method.

1. If Newton's method is used on $f(x)=x^3-2$ starting with $x_0=1$, what is x_2?
2. Devise a Newton iteration formula for computing $R^{1/3}$ where $R>0$.
3. Write a program that implements the Newton's method and find solutions accurate to within 10^{-10} for the following problems:

a. $x - 2^{-x} = 0$ for $0 \le x \le 1$

b. $e^x - x^2 + 3x - 2 = 0$ for $0 \le x \le 1$

c. $2x\cos(2x) - (x+1)^2 = 0$ for $-3 \le x \le -2$ and $-1 \le x \le 0$

d. $x\cos(x) - 2x^2 + 3x - 1 = 0$ for $0.2 \le x \le 0.3$ and $1.2 \le x \le 1.3$

Solution of nonlinear equations. Fixed point method.

1. Use Theorem to show that $g(x)=\pi + 0.5\sin(x/2)$ has a unique fixed point on $[0,2\pi]$.
2. Use Corollary to estimate the number of iterations required to achieve 10^{-4} accuracy.
3. Write a program that implements the fixed-point iteration method and find solution accurate to within 10^{-4} and compare theoretical estimate to the number actually needed.

Interpolation

1. Use appropriate Lagrange interpolating polynomial (of degree one, two and three) to approximate following data:

i	0	1	2	3
x_i	0	2	3	5
f_i	1	3	2	6

2. Construct the Lagrange interpolating polynomials for the following functions, and find a bound for the absolute error on the interval $[x_0, x_n]$:

a. $f(x)= e^{-x^2}$, $x_0=0.3$, $x_1=0.4$, $n=1$

b. $f(x)=e^{-x^2}$, $x_0=0.2$, $x_1=0.3$, $x_2=0.4$, $n=2$

a. $f(x) =e^{-x}\cos(3x)$, $x_0=0$, $x_1=0.3$, $x_2=0.6$, $n = 2$

b. $f(x)= \cos(2x)$, $x_0=0$, $x_1=0.3$, $x_2=0.6$, $n = 2$

Numerical differentiation

1. Derive the following two formulas for approximating derivatives and show that they are both $O(h^4)$ by establishing their error terms.

$$f'(x) \approx \frac{1}{12h}\left(-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)\right)$$

$$f''(x) \approx \frac{1}{12h^2}\left(-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)+f(x-2h)\right)$$

2. Derive the following two formulas for approximating the third derivative and their error terms. Which is more accurate?

$$f'''(x) \approx \frac{1}{2h^3}\left(f(x+2h)-2f(x+h)+2f(x-h)-f(x-2h)\right)$$

$$f'''(x) \approx \frac{1}{h^3}\left(f(x+3h)-3f(x+2h)+3f(x+h)-f(x)\right)$$

Numerical integration

1. Approximate the following integrals using trapezoid rule

a. $\int_0^{0.5} \frac{2}{x-4}dx$ b. $\int_1^{1.5} x^2 \ln x dx$ c. $\int_{0.5}^{1} x^4 dx$

2. Find a bound for the error in Exercise 1 using the error formula, and compare this to the actual error.

3. Repeat Exercise 1 using Simpson's rule.

4. Repeat Exercise 2 using Simpson's rule and the results of Exercise 3.

Initial-value problems for ODE

1. Use Euler's method to approximate the solutions for each of the following initial-value problems.

a. $y' = 1+(x-y)^2, \quad 2 \le x \le 3, \quad y(2)=1, \quad h=0.25$

b. $y' = 1+y/x, \quad 1 \le x \le 2, \quad y(1)=2, \quad h=0.25$

2. The actual solutions to the initial-value problems in Exercise 1 are given here. Compare the actual error at each step to the error bound.

a. $y(x) = x+1/(1-x)$

b. $y(x) = x\ln x + 2x$

3. Repeat Exercises 1 and 2 using Heun's method.

4. Repeat Exercises 1 and 2 using modified Euler's method.

5. Repeat Exercises 1 and 2 using fourth-order Runge-Kutta method.

6. Use the Euler's method and the second order Runge-Kutta method to approximate the solution of the following system of first-order differential equations

$$\begin{cases} y_1' = -4y_1 - 2y_2 + \cos x + 4\sin x, \\ y_2' = 3y_1 + y_2 - 3\sin x, \end{cases} \quad 0 \le x \le 2, \quad y_1(0)=0, y_2(0)=-1, \quad h=0.1$$

Compare the result to the actual solution

$$\begin{cases} y_1 = 2e^{-x} - 2e^{-2x} + \sin x, \\ y_2 = -3e^{-x} + 2e^{-2x}. \end{cases}$$

7. Use the Euler's method and the second order Runge-Kutta method to approximate the solution of the following higher-order differential equation

$$y'' - 2y' + y = xe^x - x,\ 0 \le x \le 1,\quad y(0) = y'(0) = 0,\quad h = 0.1$$

Compare the result to the actual solution

$$y(x) = x^3 e^x / 6 - xe^x + 2e^x - x + 2.$$

Curve fitting

1. Find the least squares polynomials of degrees 1 and 2 for the data in the following table. Compute the error E in each case.

x_i	1.0	1.1	1.3	1.5
y_i	1.84	1.96	2.21	2.45

2. Find the least squares polynomials of degrees 1, 2 and 3 for the data in the following table. Compute the error E in each case.

x_i	0	0.15	0.31	0.5	0.6	0.75
y_i	1.0	1.004	1.031	2.117	1.223	1.422

Examination questions

Question № 1

1. Lagrange interpolation polynomial. Interpolation theorem. Error analysis.
2. Using function $f(x)=\exp(2-x)$ and points

$$x_0=0,\ x_1=0.5,\ x_2=1,$$
$$f_0=7.4,\ f_1=4.5,\ f_2=2.7,$$

calculate numerically first derivative in the point x_1; find a bound for the absolute error of calculations.

Question № 2

1. Numerical differentiation. Forward-difference, backward-difference and three-point formulas for the first derivative. Error analysis.
2. Construct the Lagrange interpolating polynomial of degree 2 for the function $f(x)=\exp(2-x)$ using points

$$x_0=0,\ x_1=0.5,\ x_2=1,$$
$$f_0=7.4,\ f_1=4.5,\ f_2=2.7,$$

and find a bound for the absolute error on the interval $[x_0,x_2]$.

Question № 3

1. Numerical differentiation. Three-point formula for the second derivative. Differentiation via interpolation. Error analysis.

2. Construct the Lagrange interpolating polynomial of degree 2 for the function $f(x)=\sin(x/2)$ using points

$x_0=0$, $x_1=0.5$, $x_2=1$,

$f_0=0$, $f_1=0.2$, $f_2=0.5$,

and find a bound for the absolute error on the interval $[x_0,x_2]$.

Question № 4

1. Numerical integration. Integration via interpolation. Trapezoid rule, error analysis. Midpoint and Simpson's rule.

2. Using function $f(x)=\exp(2-x)$ and points

$$x_0=0,\ x_1=0.5,\ x_2=1,$$

$$f_0=7.4,\ f_1=4.5,\ f_2=2.7,$$

calculate numerically second derivative in the point x_1; find a bound for the absolute error of calculations.

Question № 6

1. Least squares method of interpolation. Least squares polynomial. Normal equations.

2. Use Euler's method to approximate the solution for the following initial-value problem.

$$y' = y + x,\quad 0 \le x \le 1,\quad y(0) = 1,\quad h = 0.25$$

Estimate the error bound.

Question № 7

1. Initial-value problems for ordinary differential equations. Euler's method. Error analysis. Second-order Runge-Kutta methods.

2. Find the least squares polynomial of degree 1 for the data in the following table. Compute the error of approximation.

x_i	0	1	2
y_i	1	1	2

Question № 8

1. Initial-value problems for ordinary differential equations. Second-order Runge-Kutta methods. Error analysis..

2. Find the least squares polynomial of degree 1 for the data in the following table. Compute the error of approximation.

x_i	0	1	2
y_i	1	2	2

Question № 9

1. Finite-digit arithmetic. Real data types in C++. Machine epsilon. Absolute error and relative error of calculations.

2. Find the least squares polynomial of degree 1 for the data in the following table. Compute the error of approximation.

x_i	0	1	2
y_i	2	2	1

Question № 10

1. Solution of nonlinear equations. Bisection method. Error analysis.

2. Using function $f(x)=\exp(2-x)$ and points

$$x_0=0,\ x_1=0.5,\ x_2=1,$$

$$f_0=7.4,\ f_1=4.5,\ f_2=2.7,$$

calculate numerically second derivative in the point x_1; find a bound for the absolute error of calculations.

Question № 11

1. Solution of nonlinear equations. Newton's method. Error analysis.

2. Use second order Runge-Kutta method to approximate the solution for the following initial-value problem.

$$y' = y - x,\quad 0 \le x \le 1,\quad y(0) = 1,\quad h = 0.25$$

Estimate the error bound.

References

1. Hildebrand, F. B. (1974). Introduction to Numerical Analysis (2nd edition ed.). McGraw-Hill.
2. Günther Hämmerlin and Karl-Heinz Hoffmann (1991). Numerical Mathematics, Springer-Verlag.
3. David R. Kincaid and E. Ward Cheney. (1991) Numerical analysis: mathematics of scientific computing. Brooks/Cole Publishing Company.
4. J. Stoer and R. Bulirsch (1993). Introduction to Numerical Analysis, Springer-Verlag.
5. Leader, Jeffery J. (2004). Numerical Analysis and Scientific Computation. Addison Wesley.
6. John H. Mathews and Kurtis D. Fink (2004). Numerical Methods using Matlab, 4th Ed., Pearson Prentice Hall.
7. William H. Press (2002) Numerical Recipes in C. The art of scientific computing. Cambridge University Press.

Table of contents