Syllabus of Computational Physics –402

Degree : 4th year students (physics)

Responsible Department: Physics

Academic year: 2022_2023 (2nd semester)

Teaching Languages: English

Class: 402

Course Coordinator: Dr. Abdelsalam Foad Abdelhady (A.F. Elhady)

Coordinator Email: abdelsalam.mohamed@sci.svu.edu.eg

Coordinator Office Hours: sunday : 2:4

Teaching Staff: Dr. Abdelsalam Foad

Course description: An introductory course to Computational Physics. Includes Mathematical background, algorithms and major Physical applications. Exercises using **MATLAB**

Course aims: Basic knowledge and experience in Computational Physics.

Learning outcomes - On successful completion of this module, students should be able to:

- Examine physical problem and various methods for computational solution. To implement different algorithms for solving physical problems.
- Consider various properties of numerical solution such as accuracy, stability and efficiency.

- Check correctness of computed solution.
- Examine feasibility of various options of parallel computation for solving physical problem.

Teaching arrangement and method of instruction: lecture notes for self learning lectures and exercises

Course Content:

- interpolation
- round off errors, accuracy and stability
- numerical differentiation
- numerical integration of functions
- root finding in one dimension
- solution of a set of linear equations
- Eigen vectors and Eigen values
- root finding in multi dimensions
- minimum finding
- Ordinary Differential Equations
- Partial Differential Equations
- initial value problems
- diffusion equation
- advection equation
- Monte-Carlo methods
- introduction to parallel computing

Required Reading:

Additional Reading Material:

Course/Module evaluation:

End of year written examination: 70

oral examination : 10

Midterm : 10

Project work: 5

Quizzes: 5

1. General Idea of interpolation

Suppose we are given data at discrete points only, and we wish to estimate values between these known points, Figure (1)

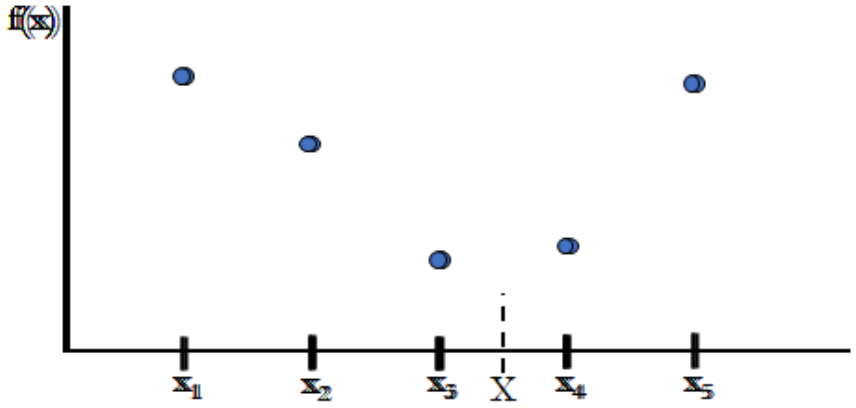


Figure 1

. The most common method used for this purpose *is fitting* or *interpolation*.

- Interpolation: passes through each datum
- Fitting: seeks to produce a curve that approximates the data, see Figure (2)

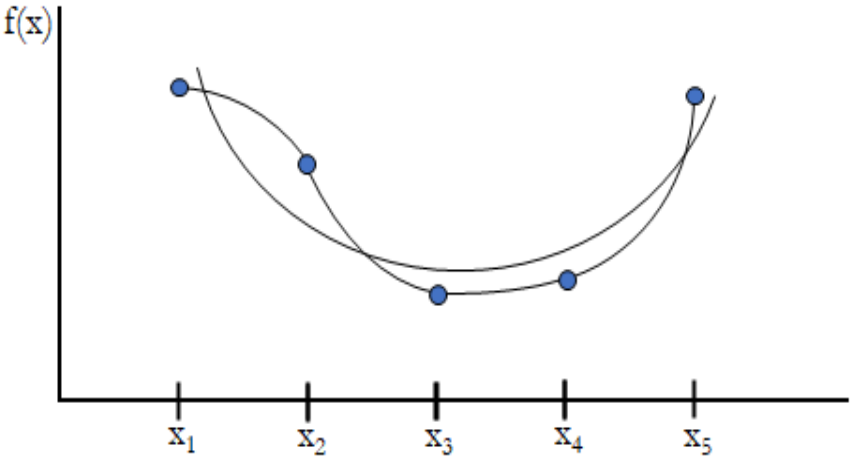


Figure 2

- **Global versus piecewise interpolation:** We can fit ***a single*** polynomial of degree n to n+1 data points, this is called global interpolation. Alternatively we can use ***a series*** of polynomials to link points together – frequently called ***splines*.**

polynomial *interpolation:* Recall that the general formula for an nth order polynomial is

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \quad (1)$$

For n+1 data points, there is one and only one polynomial of order n that passes through all the points. For example, there is only one straight line (that is, a first-order polynomial) that connects two points (figure 2.(a)). Similarly, only one parabola connects a set of three points (figure 2.(b), fig. 2(c)). Polynomial Interpolation consists of determining the unique nth order polynomial that fits n+1 data points. This polynomial then provides a formula to compute intermediate values.

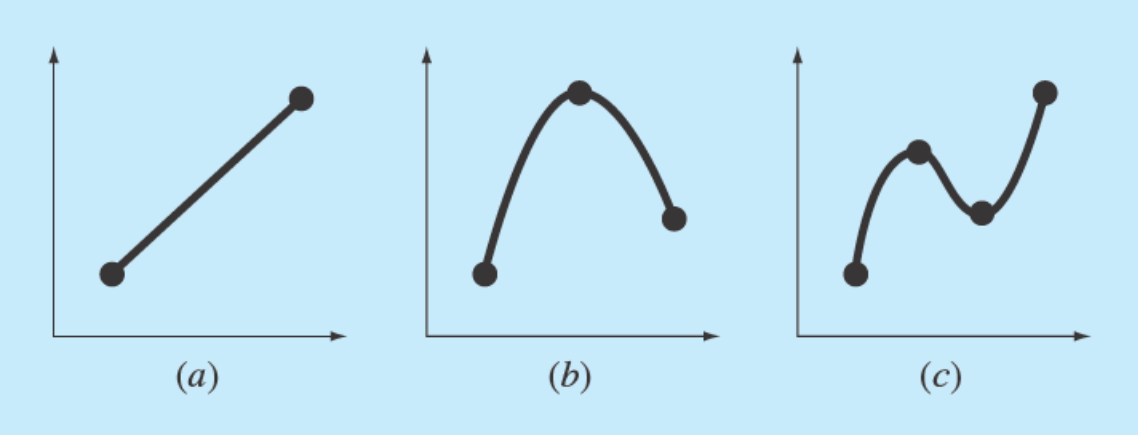


Figure 3: Examples of interpolating polynomials: (a) first-order (linear) connecting two points, (b) second-order (quadratic or parabolic) connecting three points, and (c) third-order (cubic) connecting four points.

Although there is one and only one *n*th-order polynomial that fits *n* + 1 points, there are a variety of mathematical formats in which this polynomial can be expressed. In this chapter, we will describe two

alternatives that are well-suited for computer implementation: the Newton and the Lagrange polynomials.

2. NEWTON'S DIVIDED-DIFFERENCE INTERPOLATING POLYNOMIALS

2.1.Linear Interpolation

The simplest form of interpolation is to connect two data points with a straight line. Using similar triangles,

$$\frac{f_1(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = slope \qquad (2)$$

then

$$f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \qquad (3)$$

The notation $f_1(x)$ designates that this is *a first-order interpolating polynomial*. In general, the smaller the interval between the data points, the better the approximation.

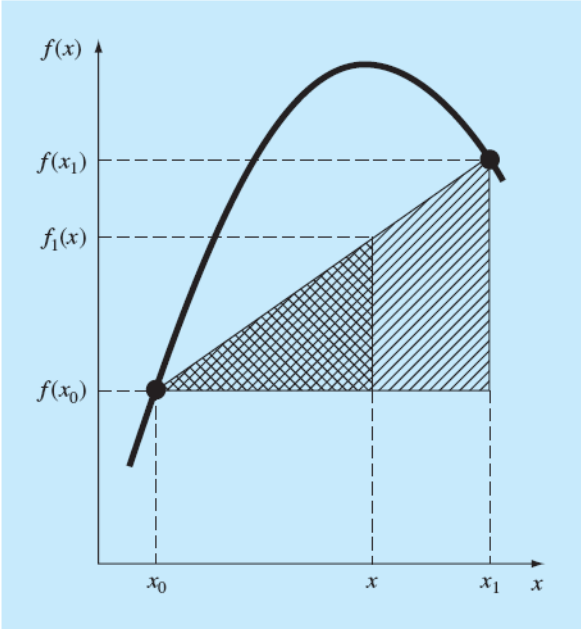


Figure 4: Graphical depiction of linear interpolation. The shaded areas indicate the similar triangles used to derive the linear-interpolation formula [Eq.2]

Example 18.1:

Estimate the natural logarithm of 2 using linear interpolation. First, perform the computation by interpolating between $\ln 1 = 0$ and $\ln 6 = 1.791759$. Then, repeat the procedure, but use a smaller interval from ln1 to ln4 (1.386294). Note that the true value of ln 2 is 0.6931472.

Solution:

We use the previous equation and a linear interpolation for ln(2) from $x_0 = 1$ to $x_1 = 6$ to give

$$f_1(2) = 0 + \frac{1.791759 - 0}{6 - 1}(2 - 1) = 0.3583519$$

Which represents an error of $\varepsilon_t = 48.3\%$. Using the smaller interval from $x_0 = 1$ to $x_1 = 4$ yields

$$f_1(2) = 0 + \frac{1.386294 - 0}{4 - 1}(2 - 1) = 0.4620981$$

Thus, using the shorter interval reduces the percent relative error to $\varepsilon_t = 33.3\%$. Both interpolations are shown in the next figure, along with the true function.

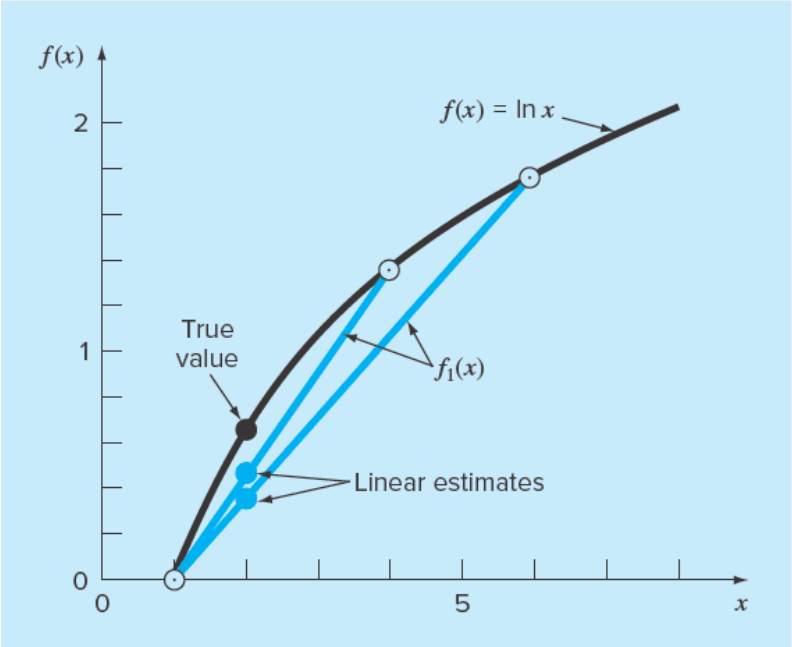


Figure 5: Two linear interpolations to estimate ln 2. Note how the smaller interval provides a better *estimate.*

2.2. Quadratic Interpolation

If three data points are available, the estimate is improved by introducing some curvature into the line connecting the points. If three data points are available, this can be accomplished with *a second-order polynomial* (also called a quadratic polynomial, or a *parabola*). A particularly convenient form for this purpose is

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) \quad (4)$$

A simple procedure can be used to determine the values of the coefficients.

$x = x_0$ $\quad b_0 = f(x_0)$

$x = x_1$ $\quad b_1 = \dfrac{f(x_1) - f(x_0)}{x - x_0}$

$x = x_2$ $\quad b_2 = \dfrac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$

Notice that, as was the case with linear interpolation, b_1 still represents the slope of the line connecting points x_0 and x_1. Thus, the first two terms of Eq. (4) are equivalent to linear interpolation from x_0 to x_1 . The last term, $b_2(x - x_0)(x - x_1)$, introduces the second-order curvature into the formula.

Example 18.2: Fit a second-order polynomial to the three points used in example 18.1:

$x_0 = 1$ $\quad f(x_0) = 0$

$x_1 = 4$ $\quad f(x_1) = 1.386294$

$x_2 = 6$ $\quad f(x_2) = 1.791759$

Use the polynomial to evaluate ln 2.

Solution. Applying Eq. (18.4) yields

$$b_0 = 0$$

Equation (18.5) yields

$$b_1 = \frac{1.386294 - 0}{4 - 1} = 0.4620981$$

and Eq. (18.6) gives

$$b_2 = \frac{\dfrac{1.791759 - 1.386294}{6 - 4} - 0.4620981}{6 - 1} = -0.0518731$$

Substituting these values into Eq. (4) yields the quadratic formula

$$f_2(x) = 0 + 0.4620981(x - 1) - 0.0518731(x - 1)(x - 4)$$

which can be evaluated at $x = 2$ to give

$$f_2(2) = 0.5658444$$

which represents a relative error of $\varepsilon_1 = 18.4\%$. Thus, the curvature introduced by the quadratic formula (Fig. 6) improves the interpolation compared with the result obtained using straight lines in Example 18.1 and Fig. 5.

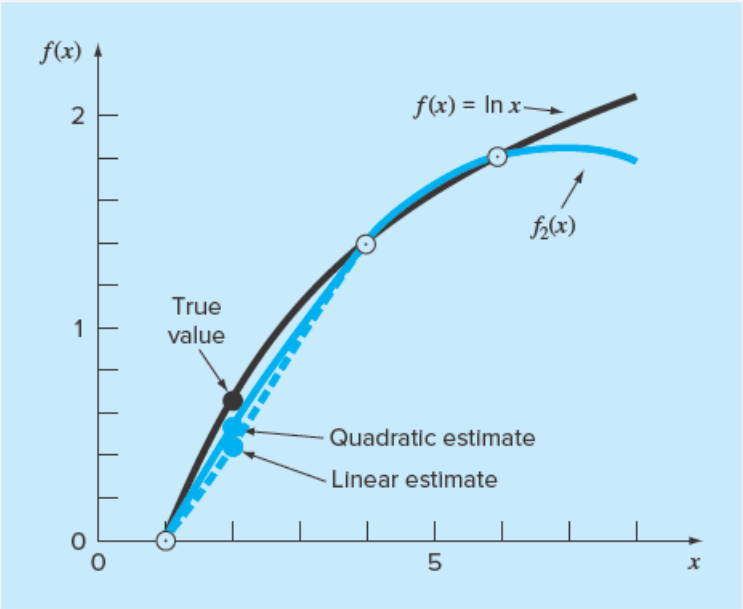


Figure 6

2.3. General Form of Newton's Interpolating Polynomials

The preceding analysis can be generalized to fit an nth order polynomial to n+1 data points. The nth-order polynomial is

$$f_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \cdots + b_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

$x = x_0 \qquad b_0 = f(x_0)$

$x = x_1 \qquad b_1 = \dfrac{f(x_1) - f(x_0)}{x - x_0}$

$x = x_2 \qquad b_2 = \dfrac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$

$x = x_{n-1} \qquad b_n = \dfrac{\dfrac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} - \cdots \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_n - x_0}$

which is called Newton's divided-difference interpolating polynomial. It should be noted that it is not necessary that the data points used in the previous equation be equally spaced or that the abscissa values necessarily be in ascending order.

Errors of Newton's Interpolating Polynomials:

Structure of interpolating polynomials is similar to the Taylor series expansion in the sense that finite divided differences are added sequentially to capture the higher order derivatives.

Example 18.3:

In Example 18.2, data points at $x_0 = 1$, $x_1 = 4$, and $x_2 = 6$ were used to estimate ln2 with a parabola. Now, adding a fourth point, [$x_3 = 5$;

$f(x_3) = 1.609438]$, estimate ln2 with a third-order Newton's interpolating polynomial

Solution

The third-order polynomial with n = 3, is

$$f_3(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2)$$

$x_0 = 1, \quad x_1 = 4, \quad x_2 = 6, \quad x_3 = 5$

$f(x_0) = \ln(1) = 0$

$f(x_1) = \ln(4) = 1.386294$

$f(x_2) = \ln(6) = 1.791759$

$f(x_3) = \ln(5) = 1.609438$

$$b_0 = f(x_0) = \ln(1) = 0$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = 1.386294$$

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} = -0.05187311$$

$$b_3 = \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_3 - x_0} == 0.007865529$$

Then

$$f_3(x) = 0 + 0.4620981\,(x - 1) - 0.05187311\,(x - 1)(x - 4) + 0.007865529\,(x - 1)(x - 4)(x - 6)$$

which can be used to evaluate $f_3(2) = 0.6287686$, which represents a relative error of $\varepsilon_t = 9.3\%$. The complete cubic polynomial is shown in the next figure.

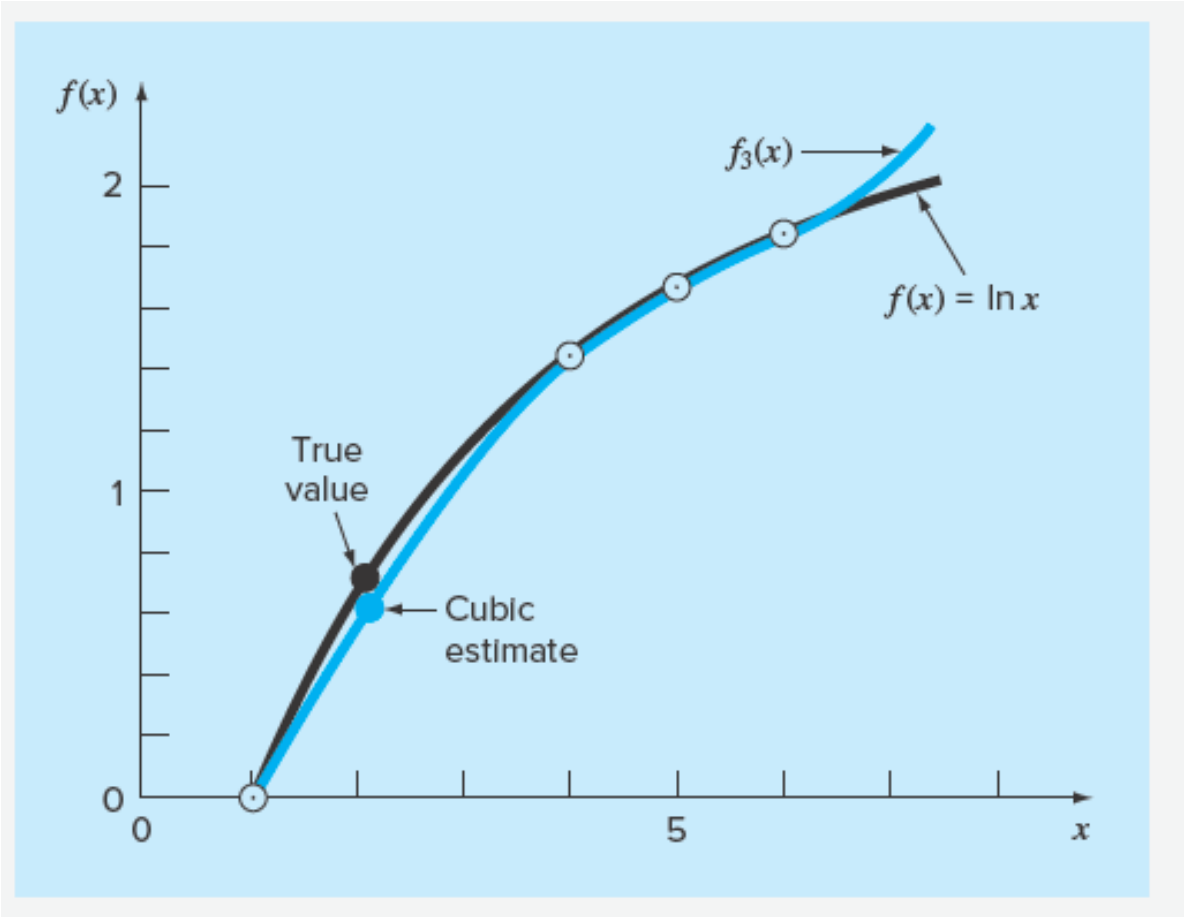


Figure 7

3. Lagrange interpolation

Lagrange interpolating polynomial can be represented concisely as

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \quad \textbf{(5)}$$

Where

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_1}{x_i - x_1} \frac{x - x_2}{x_i - x_2} \dots \frac{x - x_n}{x_i - x_n} \qquad (6)$$

where Π designates the "product of". For example, the linear (first order) version (n=1) is

$$f_1(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

$$L_0(x) = \prod_{\substack{j=0 \\ j\neq 0}}^{1} \frac{x - x_j}{x_i - x_j} = \frac{x - x_1}{x_0 - x_1}$$

$$L_1(x) = \prod_{\substack{j=0 \\ j\neq 0}}^{1} \frac{x - x_j}{x_i - x_j} = = \frac{x - x_0}{x_1 - x_0}$$

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) \quad \textbf{(7)}$$

And the second-order version is

(n=2)

$$f_2(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

$$L_0(x) = \prod_{\substack{j=0 \\ j\neq 0}}^{2} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$L_1(x) = \prod_{\substack{j=0 \\ j\neq 1}}^{2} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$, L_2(x) = \prod_{\substack{j=0 \\ j\neq 2}}^{2} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2) \quad \textbf{(8)}$$

the rationale underlying the Lagrange formulation can be grasped directly by realizing that each term $L_i(x)$ will be 1 at $x = x_i$ and 0 at all other sample points (Fig. 4). Thus, each product $L_i(x)f(x_i)$ takes on the value of $f(x_i)$ at the sample point x_i . Consequently, the summation of all the products designated by Eq. (5) is the unique *n*th-order polynomial that passes exactly through all *n* + 1 data points.

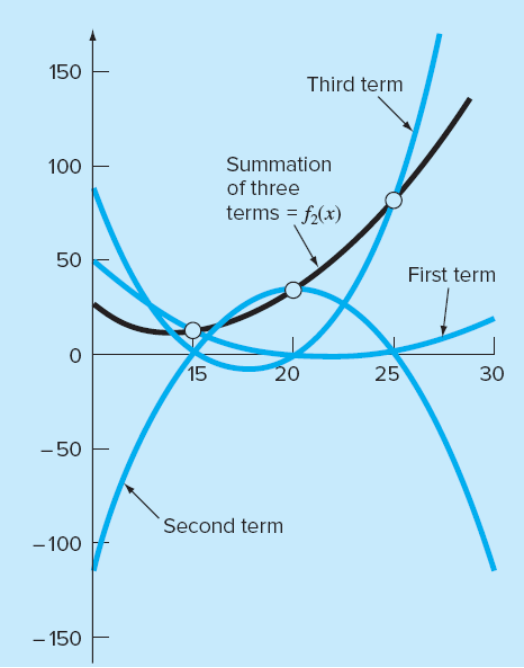


Figure 8: A visual depiction of the rationale behind the Lagrange polynomial. This figure shows a second-order case. Each of the three terms in Eq. (8) passes through one of the data points and is zero at the other two. The summation of the three terms must, therefore, be the unique second-order polynomial $f_2(x)$ that passes exactly through the three points.

Note that:

1- The factors $L_i(x)$ weights must obey the normalization rule:

$$\sum_{i=0}^{\infty} L_i(x) = 1$$

2- Lagrange interpolation just reduces to the linear interpolation described above.

3- The Lagrange interpolation has no constraints that the data be evenly spaced.

4- The difference between the value of the polynomial evaluated at some x and that of the actual function is equal to the remainder:

$$R_n = f[x, x_n, x_{n-1}, \ldots, x_0] \prod_{i=0}^{n} (x - x_i)$$

5- Lagrangian interpolation is typically used to fit only local data, though it can be use to fit the data *globally*. However, there are better ways to fit nonlinear global data ⟹polynomial least square fitting (which we are not going to have time to cover in this class).

Example 18.4 :

Lagrange Interpolating Polynomials

Problem Statement. Use a Lagrange interpolating polynomial of the first and second order to evaluate ln 2 on the basis of the data given in Example 18.2:

$x_0 = 1 \quad f(x_0) = 0$

$x_1 = 4 \quad f(x_1) = 1.386294$

$x_2 = 6 \quad f(x_2) = 1.791760$

Solution

For the first order

$$f_1(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{2 - 4}{1 - 4} \quad , \quad L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{2 - 1}{4 - 1}$$

$$f_1(2) = \frac{2 - 4}{1 - 4}0 + \frac{2 - 1}{4 - 1}1.386294 = 0.4620981$$

For the second order

$$f_2(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}\frac{x - x_2}{x_0 - x_2} = \frac{(2 - 4)}{(1 - 4)}\frac{(2 - 6)}{(1 - 6)},$$

$$L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(2 - 1)(2 - 6)}{(4 - 1)(4 - 6)}$$

$$L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(2 - 1)(2 - 4)}{(6 - 1)(6 - 4)}$$

$$f_2(2) = \frac{(2-4)(2-6)}{(1-4)(1-6)}0 + \frac{(2-1)(2-6)}{(4-1)(4-6)}1.386294 + \frac{(2-1)(2-4)}{(6-1)(6-4)}1.791760 = 0.5658444$$

3.2 Coefficients of an Interpolating Polynomial

Although "Lagrange" polynomials are well suited for determining intermediate values between points, they do not provide a polynomial in conventional form:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_x x^n$$

Since **n+1 data** points are required to determine **n+1 coefficients**, simultaneous linear systems of equations can be used to calculate "*a*"s.

$$\begin{aligned} f(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 \cdots + a_n x_0^n \\ f(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 \cdots + a_n x_1^n \\ &\vdots \\ f(x_n) &= a_0 + a_1 x_n + a_2 x_n^2 \cdots + a_n x_n^n \end{aligned}$$

Where "x"s are the knowns and "a"s are the unknowns.

Extrapolation *is the process of estimating a value of f(x) that lies outside the range of the known base points,* $x_0, x_1, \ldots, x_n$ (Fig. 9). In a previous section, we mentioned that the most accurate interpolation is usually obtained when the unknown lies near the center of the base points. Obviously, this is violated when the unknown lies outside the range, and consequently, the error in extrapolation can be very large. As depicted in Fig. 9, the open-ended nature of extrapolation represents a step into the unknown because the process extends the curve beyond the known region. As such, the true curve could easily diverge from the prediction. Extreme care should, therefore, be exercised whenever a case arises where one must extrapolate.

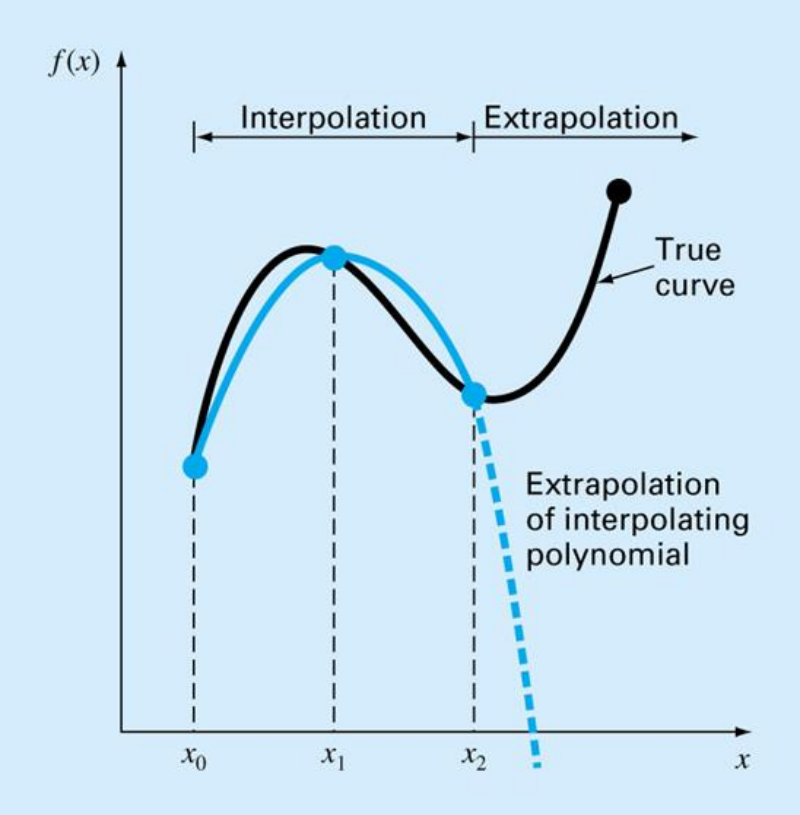


Figure 9

3.3 Problems with polynomial interpolation

I- As the number of points increases so does the degree of the polynomial. This leads to:

- Occurring of many turns and a lot of "structure" between interpolation points in high order polynomials
- High order polynomials will have a lot of "wiggles" and if points do not change smoothly they can "overshoot" in between datums.

4. Spline interpolation

Alternative approach is to apply **lower-order polynomials** to subsets of data points. Such connecting polynomials are called **spline functions.** The first order splines for a group of ordered data points can be defined as a set of linear functions:

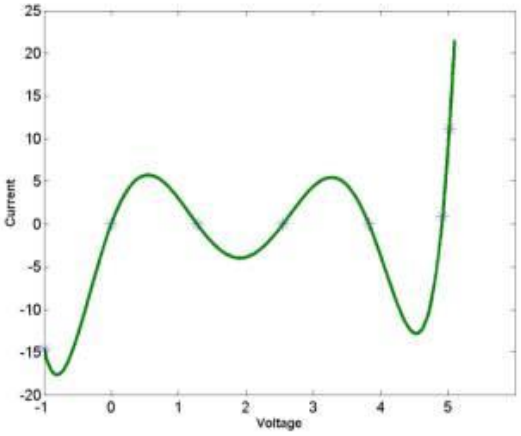


Figure 10 6th order polynomial fit

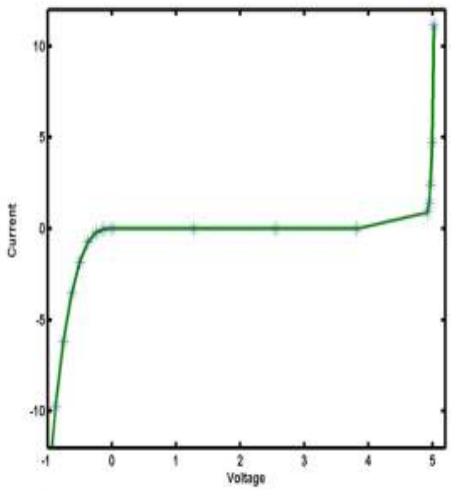


Figure 11: Piecewise linear fit

4.1 Linear spline

The first order splines for a group of ordered data points can be defined as a set of linear functions:

$$f(x) = f(x_0) + m_0(x - x_0) \qquad x_0 \le x \le x_1$$

$$f(x) = f(x_1) + m_1(x - x_1) \qquad x_1 \le x \le x_2$$

$$\vdots$$

$$f(x) = f(x_{n-1}) + m_{n-1}(x - x_{n-1}) \qquad x_{n-1} \le x \le x_n$$

$$m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Example18.8

Fit the following data with **first order splines**. Evaluate the function at x = 5.

x	f(x)
3.0	2.5
4.5	**1.0**
7.0	**2.5**
9.0	0.5

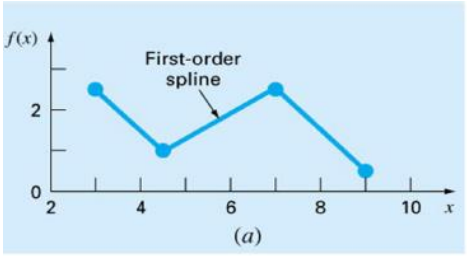


Figure 12

Solution

The data can be substituted to generate the linear spline functions. For example, for the second interval from x = 4.5 to x = 7, the function is

$$f(x) = 1.0 + \frac{2.5 - 1}{7 - 4.5}(x - 4.5)$$

$$f(5) = 1.0 + \frac{2.5 - 1}{7 - 4.5}(5 - 4.5) = 1.3$$

The equations for the other intervals can be generated, and the resulting first-order splines are plotted in Fig. 12.

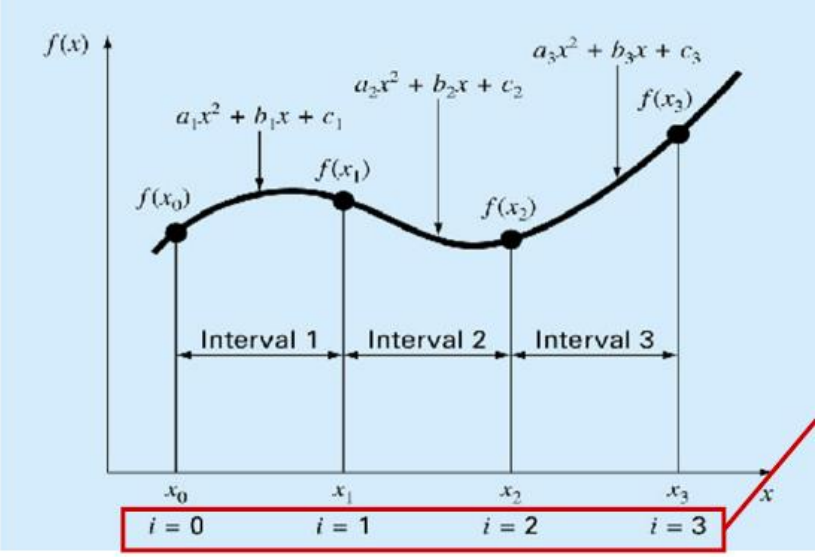
The main disadvantage of linear spline is *that they are not smooth*. At the data points where two splines meets called (a knot), the slope changes abruptly. The first derivative of the function is discontinuous at these points. This deficiency is overcome using **higher order polynomial splines** ensure smoothness at the knots by equating derivatives at these points.

4.1 Quadratic spline

Using **higher order polynomial splines** ensure smoothness at the knots by equating derivatives at these points. Because the derivation of cubic splines is somewhat involved, we have decided to first illustrate the concept of spline interpolation using second-order polynomials. These "quadratic splines" have continuous first derivatives at the knots. Although quadratic splines are not of practical importance, they serve nicely to demonstrate the general approach for developing higher-order splines.

The objective in quadratic splines is to derive a second-order polynomial for each interval between data points. The polynomial for each interval can be represented generally as

$$f_i(x) = a_i x^2 + b_i x + c \qquad (7)$$



For $i = 0, 1, 2, \ldots . n)$ there are $n + 1$ data point and n interval and $3n$ unknown constant (three, a, b,c, for each polynomial). Therefore $3n$ equations or conditions are required to evaluate the unknowns. These can be developed as follows:

1. The function values of adjacent polynomials must be equal at the interior knots, $2(n - 1)$. This condition can be written as;

$$a_{i-1}x_{i-1}^{\ 2} + b_{i-1}x_{i-1} + c_{i-1} = f_i(x_{i-1}) \quad i = 2, 3, 4, \ldots, n$$

$$a_i x_{i-1}^{\ 2} + b_i x_{i-1} + c_i = f_i(x_{i-1}) \qquad i = 2, 3, 4, \ldots, n$$

2. The first and last functions must pass through the end points, **2**.

$$a_1 x_0^{\ 2} + b_1 x_0 + c_1 = f(x_0)$$

$$a_n x_n^{\ 2} + b_n x_n + c_n = f(x_n)$$

3. The first derivatives at the interior knots must be equal **(n-1)**.

$$f'_{i-1}(x_{i-1}) = f'_i(x_{i-1})$$

$$f_i'(x) = 2a_i x + b_i$$

$$f'_{i-1}(x_{i-1}) = 2a_{i-1}x_{i-1} + b_{i-1}$$

$$f'_i(x_{i-1}) = 2a_i x_{i-1} + b_i$$

Then

$$2a_{i-1}x_{i-1} + b_{i-1} = 2a_i x_{i-1} + b_i$$

4. Assume that the second derivative is zero at the first point (the first two point will connect by straight line);

$$a_i = 0$$

Example 18.9

Fit quadratic splines to the same data employed in Example 18.8

x	3.0	4.5	7.0	9.0
f(x)	2.5	1.0	2.5	0.5

. Use the results to estimate the value of the function at *x* = 5

Solution

We have (i=0,1,2,3), there are 4 points so there are **3** intervals (n=3), **9** unknowns.

1. Equal interior points:

- **For first interior point (4.5, 1.0)**

The 1st equation:

$$x_1^2 a_1 + x_1 b_1 + c_1 = f(x_1)$$

$$(4.5)^2 a_1 + 4.5 b_1 + c_1 = f(4.5) \rightarrow \boxed{20.25 a_1 + 4.5 b_1 + c_1 = 1.0}$$

The 2nd equation:

$$x_1^2 a_2 + x_1 b_2 + c_2 = f(x_1)$$

$$(4.5)^2 a_2 + 4.5 b_2 + c_2 = f(4.5) \rightarrow \boxed{20.25 a_2 + 4.5 b_2 + c_2 = 1.0}$$

- **For second interior point (7.0, 2.5)**

The 3rd equation:

$$x_2^2 a_2 + x_2 b_2 + c_2 = f(x_2)$$

$$(7)^2 a_2 + 7 b_2 + c_2 = f(7) \rightarrow \boxed{49 a_2 + 7 b_2 + c_2 = 2.5}$$

The 4th equation:

$$x_2^2 a_3 + x_2 b_3 + c_3 = f(x_2)$$

$$(7)^2 a_3 + 7 b_3 + c_3 = f(7) \rightarrow \boxed{49 a_3 + 7 b_3 + c_3 = 2.5}$$

- First and last functions pass the end points

For the start point **(3.0, 2.5)**

$$x_0^2 a_1 + x_0 b_1 + c_1 = f(x_0) \rightarrow \boxed{9 a_1 + 3 b_1 + c_1 = 2.5}$$

For the end point **(9, 0.5)**

$$x_3^2 a_3 + x_3 b_3 + c_3 = f(x_3) \rightarrow \boxed{81 a_3 + 9 b_3 + c_3 = 0.5}$$

➢ Equal derivatives at the interior knots.

For first interior point **(4.5, 1.0)**

$$2x_1 a_1 + b_1 = 2x_1 a_2 + b_2 \rightarrow \boxed{9a_1 + b_1 = 9a_2 + b_2}$$

For second interior point **(7.0, 2.5)**

$$2x_2 a_2 + b_2 = 2x_3 a_3 + b_3 \rightarrow \boxed{14a_2 + b_2 = 14a_3 + b_3}$$

➢ Second derivative at the first point is 0

$$\boxed{f''(x_0) = a_1 = 0}$$

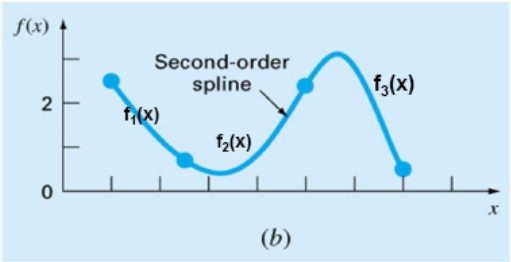
$$\begin{bmatrix} 4.5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20.25 & 4.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 49 & 7 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 49 & 7 & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 81 & 9 & 1 \\ 1 & 0 & -9 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 1 & 0 & -14 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2.5 \\ 2.5 \\ 2.5 \\ 0.5 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4.5 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.5 \end{bmatrix} \Rightarrow \begin{bmatrix} b_1 = -1 \\ c1 = 5.5 \end{bmatrix}$$

$$\begin{bmatrix} 20.25 & 4.5 & 1 \\ 49 & 7 & 1 \\ -9 & -1 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.5 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} a_2 = 0.64 \\ b_2 = -6.76 \\ c_2 = 18.46 \end{bmatrix}$$

Solving these 8 equations with 8 unknowns

$$a_1 = 0, \quad b_1 = -1, \quad c_1 = 5.5$$
$$a_2 = 0.64, \quad b_2 = -6.76, \quad c_2 = 18.46$$
$$a_3 = -1.6, \quad b_3 = 24.6, \quad c_3 = -91.3$$



$$f_1(x) = -x + 5.5, \qquad 3.0 \le x \le 4.5$$

$$f_2(x) = 0.64x^2 - 6.76x + 18.46, \qquad 4.5 \le x \le 7.0$$

$$f_3(x) = -1.6x^2 + 24.6x - 91.3, \qquad 7.0 \le x \le 9.0$$

The total quadratic spline fit is depicted in Fig. *b*. Notice that there are two shortcomings that detract from the fit:

(1) the straight line connecting the first two points and

(2) the spline for the last interval seems to swing too high. The cubic splines in the next section do not exhibit these shortcomings and, as a consequence, are better methods for spline interpolation.

4.2 Cubic Splines

Cubic splines are preferred because they provide the simplest representation that exhibits the desired appearance of smoothness.

The objective with cubic splines is to derive a third-order polynomial for each interval between knots as represented generally by

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \qquad (8)$$

Thus, for n+1 data points (i =0, 1, 2, . . . , n), there are n intervals and 4n unknown coefficients to evaluate. Consequently, 4n conditions are required for their evaluation (a's, b's ,c's and d's). These conditions are the following.

- *The function values must be equal at the interior knots (2n-2).*
- *The first and last functions must pass through the end points (2).*
- *The first derivatives at the interior knots must be equal (n-1).*
- *The second derivatives at the interior knots must be equal (n-1).*

• The second derivatives at the end knots are zero (2), (the 2nd derivative function becomes a straight line at the end points).

Once the additional end conditions are specified, we would have the 4n conditions needed to evaluate the 4n unknown coefficients.

- **Alternative technique to get Cubic Splines**

The second derivative within each interval $[x_{i-1},x_i]$ is a straight line. (the 2nd derivatives can be represented by first order Lagrange interpolating polynomials.

$$f_i''(x) = f_i''(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f_i''(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

The second derivative at any point x within the interval
The last equation can be integrated twice
2 unknown constants of integration can be evaluated by applying the boundary conditions:

1. $f(x) = f(x_{i-1})$ at x_{i-1}
2. $f(x) = f(x_i)$ at x_i

$$f_i(x) = \frac{f_i''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f_i''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 + \left[\frac{f_i(x_{i-1})}{x_i - x_{i-1}} - \frac{f_i''(x_{i-1})(x_i - x_{i-1})}{6}\right](x_i - x) + \left[\frac{f_i(x_i)}{x_i - x_{i-1}} - \frac{f_i''(x_i)(x_i - x_{i-1})}{6}\right](x - x_{i-1})$$

Unknowns:

$f''(x_i)$

$f''(x_{i-1})$

$i = 0, 1, \ldots, n$

This equation result with **n-1** unknown second derivatives where, for boundary points:

$$f''(x_o) = f''(x_n) = 0$$

Example :

Fit cubic splines to the same data used in Examples 18.8 and 18.9. Utilize the results to estimate the value of the function at $x = 5$.

Solution:

- Natural Spline:

$$f''(x_0) = f''(3) = 0, \quad f''(x_3) = f''(9) = 0$$

$$1.5f''(3)+2\times4f''(4.5)+2.5f''(7)=\frac{6}{2.5}(2.5-1)+\frac{6}{1.5}(2.5-1)$$

Since $f''(3)=0$

$$8f''(4.5)+2.5f''(7)=9.6 \quad \ldots\ldots\ldots\ldots (eq.1)$$

➢**For 2nd interior point (x_2 = 7)**

x	3.0	4.5	7.0	9.0
f(x)	2.5	1.0	2.5	0.5

$$x_i-x_{i-1}=x_2-x_1=7-4.5=2.5$$

$$x_{i+1}-x_{i-1}=x_3-x_1=9-4.5=4.5$$

$$x_{i+1}-x_i=x_3-x_2=9-7=2$$

Apply the following equation:

$$(x_i-x_{i-1})f''(x_{i-1})+2(x_{i+1}-x_{i-1})f''(x_i)+(x_{i+1}-x_i)f''(x_{i+1})$$
$$=\frac{6}{x_{i+1}-x_i}\left[f(x_{i+1})-f(x_i)\right]+\frac{6}{x_i-x_{i-1}}\left[f(x_{i-1})-f(x_i)\right]$$

$$2.5f''(4.5)+2\times4.5f''(7)+2f''(9)=\frac{6}{2}(0.5-2.5)+\frac{6}{2.5}(1-2.5)$$

Since $f''(9)=0$

$$2.5f''(4.5)+9f''(7)=-9.6 \quad \ldots\ldots\ldots\ldots (equ\ 2)$$

Solve the two equations:

$$\left.\begin{aligned}8f_i''(4.5)+2.5f_i''(7)&=9.6\\2.5f_i''(4.5)+9f_i''(7)&=-9.6\end{aligned}\right\} yeild\ f''(4.5)=1.67909,\ f''(7)=-1.53308$$

The first interval (*i=1*), apply for the equation:

$$f_i(x)=\frac{f_i''(x_{i-1})}{6(x_i-x_{i-1})}(x_i-x)^3+\frac{f_i''(x_i)}{6(x_i-x_{i-1})}(x-x_{i-1})^3$$
$$+\left[\frac{f_i(x_{i-1})}{x_i-x_{i-1}}-\frac{f_i''(x_{i-1})(x_i-x_{i-1})}{6}\right](x_i-x)+\left[\frac{f_i(x_i)}{x_i-x_{i-1}}-\frac{f_i''(x_i)(x_i-x_{i-1})}{6}\right](x-x_{i-1})$$

$$f_1(x)=0(x_i-3)^3+\frac{1.67909}{6(1.5)}(x-3)^3+\left[\frac{2.5}{1.5}-\frac{0(1.5)}{6}\right](4.5-x)+\left[\frac{1}{1.5}-\frac{1.67909(1.5)}{6}\right](x-3)$$

$$f_1(x)=0.186566(x-3)^3+1.6667\ (4.5-x)+0.24689(x-3)$$

The 2nd interval (*i =2*), apply for the equation:

$$f_2(x)=\frac{1.67909}{6(2.5)}(7-x)^3+\frac{-1.53308}{6(2.5)}(x-4.5)^3+\left[\frac{1}{2.5}-\frac{-1.67909(2.5)}{6}\right](7-x)$$
$$+\left[\frac{2.5}{2.5}-\frac{-1.53308(2.5)}{6}\right](x-4.5)$$

$$f_2(x)=0.111939(7-x)^3-0.102205\,(x-4.5)^3-0.29962(7-x)+1.638783\,(x-4.5)$$

The 3rd interval (*i =3*),

$$f_3(x)=-0.127757(9-x)^3+1.761027\,(9-x)+0.25\,(x-7)$$

For x = 5: $f_2(x)=f_2(5)=1.102886$

4.3 Matlab interpolates

Interp1

Matlab has its own interpolation routine interp1 which does the things discussed in the previous two sections automatically. Suppose you have a set of data points {x, y} and you have a different set of x-values {x_i} for which you want to find the corresponding {yi} values by interpolating in the {x, y} data set. You simply use any one of these three forms of the interp1 command:

```
yi= interp1 (x,y,xi,'linear')
yi= interp1 (x,y,xi,'pchip')
yi= interp1 (x,y,xi,'spline')
```

We haven't talked about spline interpolation yet. It is a piece-wise polynomial fit that typically does an excellent job of matching smooth functions.

Here is an example of how each of these three types of interpolation works on a crude data set representing the sine function.

Listing 10.1 (ch10ex1.m)

```
clear; close all;

% make the crude data set with dx too big for
% good accuracy
dx=pi/5;
x=0:dx:2*pi;
y=sin(x);

% make a fine x-grid
xi=0:dx/20:2*pi;

% interpolate on the coarse grid to
% obtain the fine yi values

% linear interpolation
yi=interp1(x,y,xi,'linear');

% plot the data and the interpolation
plot(x,y,'b*',xi,yi,'r-')
title('Linear Interpolation')

% cubic interpolation
yi=interp1(x,y,xi,'pchip');

% plot the data and the interpolation
figure
```

```
plot(x,y,'b*',xi,yi,'r-')
title('Cubic Interpolation')

% spline interpolation
yi=interp1(x,y,xi,'spline');

% plot the data and the interpolation
figure
plot(x,y,'b*',xi,yi,'r-')
title('Spline Interpolation')
```

4.4 Two-dimensional interpolation

Matlab also knows how to do 2-dimensional interpolation on a data set of $\{x, y, z\}$ to find approximate values of $z(x, y)$ at points $\{xi\ , yi\ \}$ which don't lie on the data points $\{x, y\}$. In the completely general situation where your data points $\{x, y, z\}$ don't fall on a regular grid, you can use the command TriScatteredInterp to interpolate your function onto an arbitrary new set of points $\{xi\ , yi\ \}$, such as an evenly spaced 2-dimensional grid for plotting. Examine the code below to see how TriScatteredInterp works, and play with the value of N and see how the interpolation quality depends on the number of points.

Listing 10.2 (ch10ex2.m)

```
clear; close all;

% Make some "data" at random points x,y points
N=200;
x = (rand(N,1)-0.5)*6;
y = (rand(N,1)-0.5)*6;
z = cos((x.^2+y.^2)/2);

% Create an interpolating function named F
F = TriScatteredInterp(x,y,z,'natural');

% Create an evenly spaced grid to interpolate onto
xe = -3:.1:3;
ye = xe;
[XE,YE] = ndgrid(xe,ye);

% Evaluate the interpolation function on the even grid
ZE = F(XE,YE);

% plot the interpolated surface
surf(XE,YE,ZE);

% overlay the "data" as dots
hold on;
plot3(x,y,z,'.');
axis equal
```

The TriScatteredInterp command is very powerful in the sense that you can ask it to estimate $z(x, y)$ for arbitrary x and y (within your data range). However, for large data sets it can be slow. In the case that your data set is already on a regular grid, its much faster to use the interpn command, like this:

Listing 10.3 (ch10ex3.m)

```
clear; close all;

x=-3:.4:3; y=x;

% build the full 2-d grid for the crude x and y data
% and make a surface plot
[X,Y]=ndgrid(x,y);
Z=cos((X.^2+Y.^2)/2);
surf(X,Y,Z);
title('Crude Data')

% now make a finer 2-d grid, interpolate linearly to
% build a finer z(x,y) and surface plot it.

% Because the interpolation is linear the mesh is finer
% but the crude corners are still there
xf=-3:.1:3;
yf=xf;
[XF,YF]=ndgrid(xf,yf);
ZF=interpn(X,Y,Z,XF,YF,'linear');
figure
surf(XF,YF,ZF);
title('Linear Interpolation')

% Now use cubic interpolation to round the corners.  Note that
% there is still trouble near the edge because these points
% only have data on one side, so interpolation doesn't work well

ZF=interpn(X,Y,Z,XF,YF,'cubic');
figure
surf(XF,YF,ZF);
title('Cubic Interpolation')

% Now use spline interpolation to also round the corners and
% see how it is different from cubic.  You should notice that
% it looks better, especially near the edges.  Spline
% interpolation is often the best.

ZF=interpn(X,Y,Z,XF,YF,'spline');
figure
surf(XF,YF,ZF);
title('Spline Interpolation')
```

In this example our grids were created using **ndgrid**. If you choose to use the meshgrid command to create your grids, you'll need to use the command interp2 instead of **intern.**

Chapter2: Numerical Integration and Differentiation

PART I: Numerical Integration

PART I: Numerical Integration

Newton-Cotes Integration Formulas

The idea of Newton-Cotes formulas is to replace a complicated function or tabulated data with an approximating function that is easy to integrate

$$I = \int_a^b f(x)dx \approx \int_a^b f_n(x)dx$$

where $f_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$.

1. The Trapezoidal Rule

Using the first order Taylor series to approximate *f*(*x*),

$$I = \int_a^b f(x)dx \approx \int_a^b f_1(x)dx$$

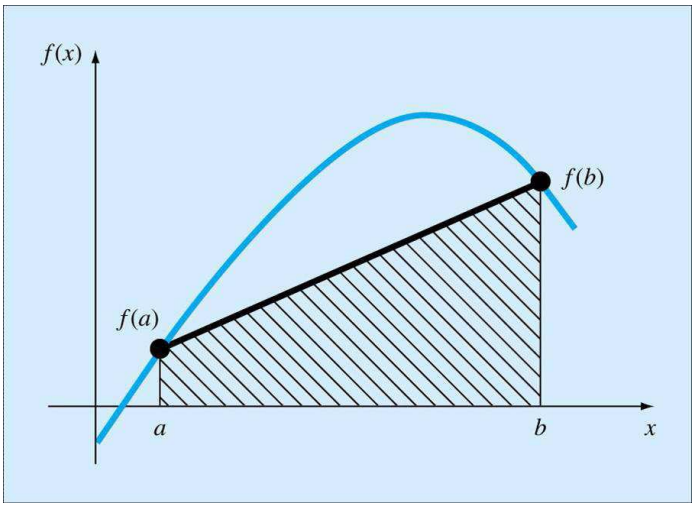
where

$$f_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

Then

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

$$I = (b - a)\frac{f(a) + f(b)}{2}$$



The trapezoidal rule is equivalent to approximating the area of the trapezoidal under the straight line connecting *f*(*a*) and *f*(*b*). An estimate for the local truncation error of a single application of the trapezoidal rule can be obtained using Taylor series as

$$E_t = -\frac{1}{12} f''(\xi)(b-a)^3$$

where ξ is a value between *a* and *b*.

Example: Use the trapezoidal rule to numerically integrate

$$f(x) = 0.2 + 25x$$

from $a = 0$ to $b = 2$.

Solution: $f(a) = f(0) = 0.2$, and $f(b) = f(2) = 50.2$.

$$I = (b-a)\frac{f(b)+f(a)}{2} = (2-0) \times \frac{0.2+50.2}{2} = 50.4$$

The true solution is

$$\int_0^2 f(x)dx = (0.2x + 12.5x^2)|_0^2 = (0.2 \times 2 + 12.5 \times 2^2) - 0 = 50.4$$

Because $f(x)$ is a linear function, using the trapezoidal rule gets the exact solution.

Example: Use the trapezoidal rule to numerically integrate

$$f(x) = 0.2 + 25x + 3x^2$$

from $a = 0$ to $b = 2$.

Solution: $f(0) = 0.2$, and $f(2) = 62.2$.

$$I = (b-a)\frac{f(b)+f(a)}{2} = (2-0) \times \frac{0.2+62.2}{2} = 62.4$$

The true solution is

$$\int_0^2 f(x)dx = (0.2x + 12.5x^2 + x^3)|_0^2 = (0.2 \times 2 + 12.5 \times 2^2 + 2^3) - 0 = 58.4$$

The relative error is

$$|\epsilon_t| = \left|\frac{58.4 - 62.4}{58.4}\right| \times 100\% = 6.85\%$$

Multiple-application trapezoidal rule:

Using smaller integration interval can reduce the approximation error. We can divide the integration interval from *a* to *b* into a number of segments and apply the trapezoidal rule to each segment. Divide (*a; b*) into *n* segments of equal width. Then

$$I = \int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx$$

where $a = x_0 < x_1 < \ldots < x_n = b$, and $x_i - x_{i-1} = h = \frac{b-a}{n}$, for $i = 1, 2, \ldots, n$. Substituting the Trapezoidal rule for each integral yields

$$I \approx h\frac{f(x_0) + f(x_1)}{2} + h\frac{f(x_1) + f(x_2)}{2} + \ldots + h\frac{f(x_{n-1}) + f(x_n)}{2}$$

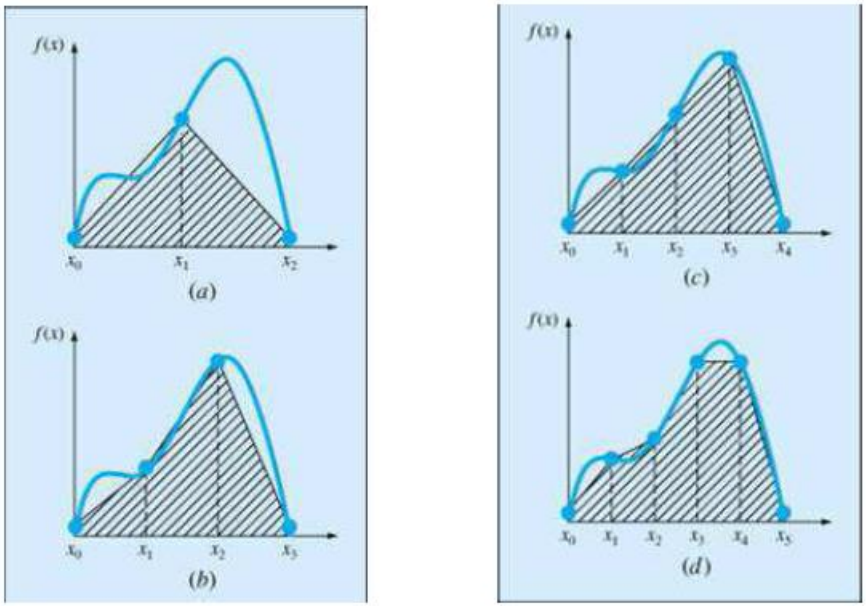
$$I = \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

The approximation error using the multiple trapezoidal rule is a sum of the individual errors, i.e.,

$$E_t = -\sum_{i=1}^{n} \frac{h^3}{12} f''(\xi_i) = -\sum_{i=1}^{n} \frac{(b-a)^3}{12n^3} f''(\xi_i)$$

Let $\overline{f''} = \frac{\sum_{i=1}^{n} f''(\xi_i)}{n}$. Then the approximate error is

$$E_t = -\frac{(b-a)^3}{12n^2}\overline{f''}$$



Example: Use the 2-segment trapezoidal rule to numerically integrate

$$f(x) = 0.2 + 25x + 3x^2$$

from $a = 0$ to $b = 2$.

Solution: $n = 2$, $h = (a - b)/n = (2 - 0)/2 = 1$.

$f(0) = 0.2$, $f(1) = 28.2$, and $f(2) = 62.2$.

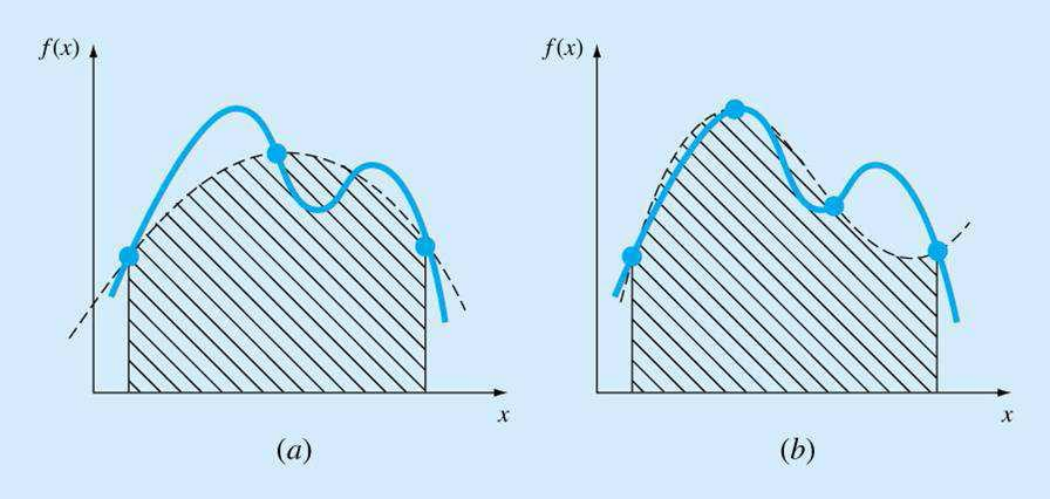
$$I = (b - a)\frac{f(0) + 2f(1) + f(2)}{2n} = 2 \times \frac{0.2 + 2 \times 28.2 + 62.2}{4} = 59.4$$

The relative error is

$$|\epsilon_t| = \left|\frac{58.4 - 59.4}{58.4}\right| \times 100\% = 1.71\%$$

2. Simpson's Rules

Aside from using the trapezoidal rule with finer segmentation, another way to improve the estimation accuracy is to use higher order polynomials.



Simpson's 1/3 rule:

Given function values at 3 points as (x_0; $f(x_0)$), (x_1; $f(x_1)$), and (x_2; $f(x_2)$), we can estimate $f(x)$ using Lagrange polynomial interpolation. Then

$$I = \int_a^b f(x)dx \approx \int_{x_0}^{x_2} \left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \right] dx$$

When $a = x_0$, $b = x_2$, $(a+b)/2 = x_1$, and $h = (b-a)/2$,

$$I = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

It can be proved that single segment application of Simpson's 1=3 rule has a truncation error of

$$E_t = -\frac{1}{90} h^5 f^{(4)}(\xi)$$

Where ξ is between a and b.

Simpson's 1/3 rule yields exact results for third order polynomials even though it is derived from parabola.

Example: Use Simpson's 1/3 rule to integrate

$$f(x) = 0.2 + 25x + 3x^2 + 8x^3$$

from $a = 0$ to $b = 2$.

Solution: $f(0) = 0.2$, $f(1) = 36.2$, and $f(2) = 126.2$.

$$I = (b-a)\frac{f(0) + 4f(1) + f(2)}{6} = 2 \times \frac{0.2 + 4 \times 36.2 + 126.2}{6} = 90.4$$

The exact integral is

$$\int_0^2 f(x)dx = (0.2x+12.5x^2+x^3+2x^4)|_0^2 = (0.2\times2+12.5\times2^2+2^3+2\times2^4)-0 = 90.4$$

Example: Use Simpson's $1/3$ rule to integrate

$$f(x) = 0.2 + 25x + 3x^2 + 2x^4$$

from $a = 0$ to $b = 2$.

Solution: $f(0) = 0.2$, $f(1) = 30.2$, and $f(2) = 94.2$.

$$I = (b-a)\frac{f(0) + 4f(1) + f(2)}{6} = 2 \times \frac{0.2 + 4 \times 30.2 + 94.2}{6} = 71.73$$

The exact integral is

$$\int_0^2 f(x)dx = (0.2x+12.5x^2+x^3+0.4x^5)|_0^2 = (0.2\times2+12.5\times2^2+2^3+0.4\times2^5)-0 = 71.2$$

The relative error is

$$|\epsilon_t| = \left|\frac{71.2 - 71.73}{71.2}\right| = 0.7\%$$

- **Multiple-application Simpson's 1/3 rule**

Dividing the integration interval into n segments of equal width, we have

$$I = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots + \int_{x_{n-2}}^{x_n} f(x)dx$$

where $a = x_0 < x_1 < \ldots < x_n = b$, and $x_i - x_{i-1} = h = (b-a)/n$, for $i = 1, 2, \ldots, n$. Substituting the Simpson's $1/3$ rule for each integral yields

$$I \approx 2h\frac{f(x_0)+4f(x_1)+f(x_2)}{6}+2h\frac{f(x_2)+4f(x_3)+f(x_4)}{6}$$
$$+\cdots+2h\frac{f(x_{n-2})+4f(x_{n-1})+f(x_n)}{6}$$

$$I=\frac{h}{3}\left[f(x_0)+4\sum_{i=1,3,\ldots}^{n-1}f(x_i)+2\sum_{i=2,4,\ldots}^{n}f(x_i)++f(x_n)\right]$$

Example: Use 4-segment Simpson's $1/3$ rule to integrate

$$f(x)=0.2+25x+3x^2+2x^4$$

from $a=0$ to $b=2$.

Solution: $n=4$, $h=(b-a)/n=0.5$.

$f(x_0)=f(0)=0.2$, $f(x_1)=f(0.5)=13.575$, $f(x_2)=f(1)=30.2$, $f(x_3)=f(1.5)=54.575$, and $f(x_4)=f(2)=94.2$.

$$I=\frac{h}{3}\left[f(x_0)+4\sum_{i=1,3,}^{n-1}f(x_i)+2\sum_{i=2,4}^{n}f(x_i)+f(x_n)\right]$$

$$I=\frac{h}{3}[f(x_0)+f(x_n)+4[f(x_1)+f(x_3)]+2f(x_2)]$$

$$I=\frac{0.5}{3}[0.2+94.2+4[13.575+54.575]+2\times30.2]=71.2333$$

The exact integral is

$$\int_0^2 f(x)dx=(0.2x+12.5x^2+x^3+0.4x^5)|_0^2=(0.2\times2+12.5\times2^2+2^3+0.4\times2^5)-0=71.2$$

The relative error is

$$|\epsilon_t|=\left|\frac{71.2-71.2333}{71.2}\right|=0.047\%$$

Simpson's 3/8 rule

This is to use a third-order Lagrange polynomial to fit to four points of $f(x)$ and yields

$$I = \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$$

where $h = (b-a)/3$. The approximation error using this rule is

$$E_t = -\frac{3}{80}h^5 f^{(4)}(\xi) = -\frac{(b-a)^5}{6480}f^{(4)}(\xi)$$

where ξ is between a and b.

3 Integration of Equations

Newton-Cotes algorithms for equations

Compare the following two Pseudocodes for multiple applications of the trapezoidal rule.

Pseudocode 1: Algorithm for multiple applications of the trapezoidal rule

```
function Trapm(h,n,f)
  sum=f0
  for i=1:n-1
    sum=sum+2*fi
  end
  sum=sum+fn
  Trapm=h*sum/2
```

Pseudocode 2: Algorithm for multiple application of the trapezoidal rule when function $f(x)$ is available

```
function TrapEq(n,a,b)
  h=(b-a)/n
  x=a
  sum=f(x)
  for i=1:n-1
```

```
    x=x+h
    sum=sum+2*f(x)
  end
  sum=sum+f(b)
  TraEq=(b-a)*sum/(2*n)
```

Pseudocode 1 can be used when only a limited number of points are given or the function is available. Pseudocode 2 is for the case where the analytical function is available. The difference between the two pseudocodes is that in Pseudocode 2 neigher the independent nor the dependent variable values are passed into the function via its argument as in Pseudocode 1. When the analytical function is available, the function values are computed using calls to the function being analyzed, $f(x)$.

PART II: Numerical differentiation:

Finite Divided Difference

First Order Derivatives:

The first forward finite divided difference

Using Taylor series,

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + O(h^3)$$

where $h = x_{i+1} - x_i$. Then $f'(x_i)$ can be found as

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

The first forward finite divided difference is

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h}$$

The first backward finite divided difference

Using Taylor series,

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + O(h^3)$$

where $h = x_i - x_{i-1}$. Then $f'(x_i)$ can be found as

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$$

and $f'(x_i)$ can also be approximated as

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{h}$$

which is called the first backward finite divided difference.

- The first centered finite divided difference

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + O(h^3)$$

and $f'(x_i)$ can be found as

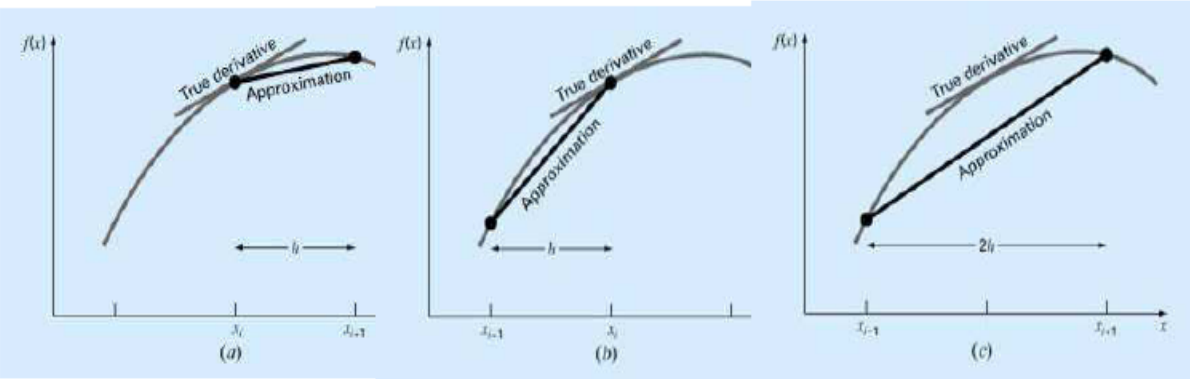
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2)$$

and $f'(x_i)$ can also be approximated as

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

which is called the first centered finite divided difference.

Notice that the truncation error is of the order of h^2 in contrast to the forward and backward approximations that are of the order of h. Therefore, the centered difference is a more accurate representation of the derivative.



Graphical depiction of (a) forward, (b) backward, and (c) centered finite-divided-difference approximations of the first derivative

Example: Estimate the first derivative of

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

at $x = 0.5$ using a step size $h = 0.5$. Repeat the computation using $h = 0.25$.

Solution:

The problem can be solved analytically

$$f'(x) = -0.4x^3 - 0.45x^2 - x - 0.25$$

and $f'(0.5) = -0.9125$.

When $h = 0.5$, $x_{i-1} = x_i - h = 0$, and $f(x_{i-1}) = 1.2$; $x_i = 0.5$, $f(x_i) = 0.925$; $x_{i+1} = x_i + h = 1$, and $f(x_{i+1}) = 0.2$.

The forward divided difference:

$$f'(0.5) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{0.2 - 0.925}{0.5} = -1.45$$

The percentage relative error:

$$|\epsilon_t| = \left|\frac{(-0.9125) - (-1.45)}{-0.9125}\right| \times 100\% = 58.9\%$$

The backward divided difference:

$$f'(0.5) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{0.925 - 1.2}{0.5} = -0.55$$

The percentage relative error:

$$|\epsilon_t| = \left|\frac{(-0.9125) - (-0.55)}{-0.9125}\right| \times 100\% = 39.7\%$$

The centered divided difference:

$$f'(0.5) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} = \frac{0.925 - 1.2}{2 \times 0.5} = -1.0$$

The percentage relative error:

$$|\epsilon_t| = \left|\frac{(-0.9125) - (-1.0)}{-0.9125}\right| \times 100\% = 9.6\%$$

When $h = 0.25$, $x_{i-1} = x_i - h = 0.25$, and $f(x_{i-1}) = 1.1035$; $x_i = 0.5$, $f(x_i) = 0.925$; $x_{i+1} = x_i + h = 0.75$, and $f(x_{i+1}) = 0.6363$.

The forward divided difference:

$$f'(0.5) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{0.6363 - 0.925}{0.25} = -1.155$$

The percentage relative error:

$$|\epsilon_t| = \left|\frac{(-0.9125) - (-1.155)}{-0.9125}\right| \times 100\% = 26.5\%$$

The backward divided difference:

$$f'(0.5) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{0.925 - 1.1035}{0.25} = -0.714$$

The percentage relative error:

$$|\epsilon_t| = \left| \frac{(-0.9125) - (-0.714)}{-0.9125} \right| \times 100\% = 21.7\%$$

The centered divided difference:

$$f'(0.5) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} = \frac{0.6363 - 1.1035}{2 \times 0.25} = -0.934$$

The percentage relative error:

$$|\epsilon_t| = \left| \frac{(-0.9125) - (-0.934)}{-0.9125} \right| \times 100\% = 2.4\%$$

Using centered finite divided difference and small step size achieves lower approximation error.

Higher Order Derivatives:

- The second forward finite divided difference

$$f(x_{i+2}) = f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2!}(2h)^2 + O(h^3) \quad (2)$$

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + O(h^3) \quad (3)$$

(2)-(3)$\times$2:

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 + O(h^3)$$

$f''(x_i)$ can be found as

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h)$$

and $f''(x_i)$ can be approximated as

$$f''(x_i) \approx \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2}$$

This is the second forward finite divided difference.

- The second backward finite divided difference

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} + O(h)$$

and $f''(x_i)$ can be approximated as

$$f''(x_i) \approx \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2}$$

is the second backward finite divided difference.

- The second centered finite divided difference

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + O(h^4) \quad (4)$$

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f^{(3)}(x_i)}{3!}h^3 + O(h^4) \quad (5)$$

(4)+(5):

$$f(x_{i+1}) + f(x_{i-1}) = 2f(x_i) + f''(x_i)h^2 + O(h^4) \quad (6)$$

Then $f''(x_i)$ can be solved from (6) as

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2)$$

and

$$f''(x_i) \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$

is the second centered finite divided difference.

High-Accuracy Numerical Differentiation

- The second forward finite divided difference

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(x_i)}{2}h + O(h^2) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \quad (7)$$

$$f''(x) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h) \quad (8)$$

Substitute (8) into (7),

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{2h} + O(h^2) \quad (9)$$

Then we have

$$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h} + O(h^2) \quad (10)$$

- The second backward finite divided difference

$$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h} + O(h^2) \quad (11)$$

- The second centered finite divided difference

$$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h} + O(h^4) \quad (12)$$

Example: $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$, $x_i = 0.5$, $h = 0.25$.

$x_i = 0.5$, $x_{i-1} = x_i - h = 0.25$, $x_{i-2} = 0$, $x_{i+1} = x_i + h = 0.75$, $x_{i+2} = 1$.
$f(x_i) = 0.925$, $f(x_{i-1}) = 1.1035$, $f(x_{i-2}) = 1.2$, $f(x_{i+1}) = 0.6363$, and $f(x_{i+2}) = 0.2$.
Using the forward f.d.d., $f'(x_i) \doteq -1.155$, $\epsilon_t = -26.5\%$
Using the backward f.d.d., $f'(x_i) \doteq 0.714$, $\epsilon_t = 21.7\%$
Using the centered f.d.d., $f'(x_i) \doteq -0.934$, $\epsilon_t = -2.4\%$
Using the second forward f.d.d.,
$f'(x_i) = \frac{-f(x_{i+2})+4f(x_{i+1}-3f(x_i)}{2h} = \frac{-0.2+4\times0.6363-3\times0.925}{2\times0.25} = -0.8594$
$\epsilon_t = \left|\frac{-0.8594-(-0.9125)}{-0.9125}\right| \times 100\% = 5.82\%$
Using the second backward f.d.d., $f'(x_i) = \frac{3f(x_i)-4f(x_{i-1}+f(x_{i-2})}{2h} = -0.8781$, $\epsilon_t = 3.77\%$

Using the second centered f.d.d., $f'(x_i) = \frac{-f(x_{i+2})+8f(x_{i+1}-8f(x_{i-1})+f(x_{i-2})}{12h} = -0.9125$, $\epsilon = 0\%$.

Richardson Extrapolation

This is to use two derivative estimates to compute a third, more accurate one.

$$D \approx \frac{4}{3}D(h_2) - \frac{1}{3}D(h_1),\ h_2 = \frac{h_1}{2} \quad (13)$$

Example: $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$, $x_i = 0.5$, $h_1 = 0.5$, $h_2 = 0.25$.
Solution:
With h_1, $x_{i+1} = 1$, $x_{i-1} = 0$, $D(h_1) = \frac{f(x_{i+1})-f(x_{i-1})}{2h_1} = \frac{0.2-1.2}{1} = -1.0$, $\epsilon_t = -9.6\%$.
With h_2, $x_{i+1} = 0.75$, $x_{i-1} = 0.25$, $D(h_2) = \frac{f(x_{i+1})-f(x_{i-1})}{2h_2} = -0.934375$, $\epsilon_t = -2.4\%$.

$D = \frac{4}{3}D(h_2) - \frac{1}{3}D(h_1) = \frac{4}{3} \times (-0.934575) - \frac{1}{3} \times (-1) = -0.9125$, $\epsilon_t = 0$.
For centered difference approximations with $O(h^2)$, using (13) yields a new estimate of $O(h^4)$.

3- Differential Equations

The following are some differential equations:

$$y' = (x^2 + y)e^x$$

$$y'' = y'x + xy^2$$

$$xy''' + (1 - x^2)yy'' + y = (x^2 - 1)e^{\mu}$$

Def. ***Order of differential equation***: If $y^{(k)}$ is the highest order derivative in a differential equation, the equation is said to be k^{th} order differential equation.

Def. ***A solution to the differential equation***: is the value of y which satisfies the differential equation.

Example:

Consider the differential equation: $y'' = 6x + 4$

This is a second order differential equation. The function:

$$y = x^3 + 2x^2 - 1$$

satisfies the differential equation, hence, $y = x^3 + 2x^2 - 1$ is a solution to the differential equation.

Numerical Solution

Consider the equation: $y'' = 6x + 4$

A solution is $y = x^3 + 2x^2 - 1$, however, instead of writing the solution as a function of x, we can find the numerical values of y for various pivotal values of x. The solution from $x = 0$ to $x = 1$ can be expressed as follows:

x	0	0.2	0.4	0.6	0.8	1.0
y	-1	-0.912	-0.616	0.064	0.792	2.0

The values are got by the function $y = x^3 + 2x^2 - 1$. This table of numerical values of y is said to be a numerical solution to the differential equation.

The initial Value problem

Consider the differential equation: $y' = f(x, y); y(x_0) = y_0$

This is a first order differential equation. Here, the y value at $x_0 = y_0$. The solution $y\ at\ x_0$ is given, We must assume a small increment$h$. i.e.

$x_1 = x_0 + h$

$x_2 = x_1 + h$

$x_{i+1} = x_i + h$

y_0	y_1 =?	y_2 =?	y_3 =?	y_4 =?
x_0	x_1	x_2	x_3	x_4

Let us denote the y values at $x_1, x_2, \ldots \ldots$ as $y_1, y_2 \ldots$. respectively.$y_0$is given and so we must find $y_1, y_2 \ldots$. This differential equation is called an initial value problem.

Euler's Method

Consider the initial value problem:

$$y' = f(x, y);\ y(x_0) = y_0$$

y is a function of x , so we shall write the function as $y(x)$

Using Taylor series expansion

$$y(x_0 + h) = y(x_0) + \frac{h}{1!}y'(x_0) + \frac{h^2}{2!}y''(x_0) + \cdots \ldots$$

Here, $y(x_0 + h)$ denotes y value at $x_0 + h$

$y'(x_0)$denotesy' value at $x_0 + h$, e.t.c.

Given;

$$y(x_0) = y_0$$

$$y(x_0 + h) = y(x_1) = y_1 \text{ (say)}$$

$$y'(x_0) = y' \; at \; x_0$$

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

$$\rightarrow \quad y'(x_0) = f(x_0, y_0)$$

Now, let

$$y'(x_0) = f(x_0, y_0) = f_0$$

$hence, y'(x_0) = f_0$

Therefore, Taylor's expansion ***up to the first order term***, gives

$y_1 = y_0 + hf_0$

Similarly, we can derive:

$y_2 = y_1 + hf_1$

$y_3 = y_2 + hf_2$

In general,

$$y_{i+1} = y_i + hf_i$$

This is called the Euler's formula to solve an initial value problem.

Algorithm for Euler's method

1. $Define\ f(x,y)$
2. $Read\ x_0, y_0, n, h$
3. $for\ i = 0\ to\ n-1\ Do$
4. $x_{i+1} = x_i + h$
5. $y_{i+1} = y_i + hf(x_i, y_i)$
6. $Print\ x_{i+1}\ , y_{i+1}$
7. $next\ i$
8. End

Assignment: Implement the above in any programming language

Example:

Solve the initial value problem: $y' = x^2 + y^2; y(1) = 0.8;\ \ x = 1(0.5)3$

Solution

Given: $f(x,y) = x^2 + y^2, x_0 = 1, y_0 = 0.8, h = 0.5, x = 1\ to\ 3$

$y_0 = 0.8$	y_1 =?	y_2 =?	y_3 =?	y_4 =?
$x_0 = 1$	$x_1 = 1.5$	$x_2 = 2$	$x_3 = 2.5$	$x_4 = 3$

$y_1 = y_0 + hf_0$

But $f_0 = f(x_0, y_0) = f(1, 0.8) = 1.64$

Therefore, $y_1 = 0.8 + (0.5)(1.64) = 1.62$

$y_2 = y_1 + hf_1$

But $f_1 = f(x_1, y_1) = f(1.5, 1.62) = 4.8744$

Therefore, $y_2 = 1.62 + (0.5)(4.8744) = 4.0572$

$y_3 = y_2 + hf_2$

But $f_2 = f(x_2, y_2) = f(2, 4.0572) = 20.460871$

Therefore, $y_3 = 4.0572 + (0.5)(20.460871) = 14.287635$

$y_4 = y_3 + hf_3$

But $f_3 = f(x_3, y_3) = f(2.5, 14.287635) = 210.38651$

Therefore, $y_4 = 14.287635 + (0.5)(210.38651) = 119.48088$

So the numerical solution got by Euler's method is:

$y_0 = 0.8$	$y_1 = 1.62$	$y_2 = 4.0572$	$y_3 = 14.287635$	$y_4 = 119.48088$
$x_0 = 1$	$x_1 = 1.5$	$x_2 = 2$	$x_3 = 2.5$	$x_4 = 3$

Assignment: Using Euler's method, solve: $5\frac{dy}{dx} = 3x^3y; y(0) = 1$

For the interval $0 \leq x \leq 0.3, with\ h = 0.1$

Backward Euler's Method

The formula for backward Euler's method is given by:

$$y_{i+1} = y_i + hf_{i+1}$$

Where, $f_{i+1} = f(x_{i+1}, y_{i+1})$

For example, consider the initial value problem:

$$y' = 2x^3y; y(0) = 1; x = 0(0.2)0.4$$

Solution

$$f(x, y) = 2x^3y$$

$$x_0 = 0, y_0 = 1, h = 0.2$$

The backward Euler's method formula is: $y_{i+1} = y_i + hf_{i+1}$

$$\rightarrow \quad y_{i+1} = y_i + h(2x_{i+1}^3 * y_{i+1})$$

Therefore;

$$y_{i+1} - 2hx_{i+1}^3 * y_{i+1} = y_i$$

Hence

$$y_i = y_{i+1}(1 - 2hx_{i+1}^3)$$

Or

$$y_{i+1} = \frac{y_i}{(1 - 2hx_{i+1}^3)}$$

$y_0 = 1$	$y_1 = ?$	$y_2 = ?$
$x_0 = 0$	$x_1 = 0.2$	$x_2 = 0.4$

Now, put $i = 0$ in the formula:

$$y_1 = \frac{y_0}{(1 - 2hx_1^3)} = \frac{1}{1 - 2(0.2)(0.2)^3} = 1.0032102$$

Put $i = 1$ in the formula:

$$y_2 = \frac{y_1}{(1 - 2hx_2^3)} = \frac{1.0032102}{1 - 2(0.2)(0.4)^3} = 1.0295671$$

Therefore, the numerical solution to the problem is:

$y_0 = 1$	$y_1 = 1.0032102$	$y_2 = 1.0295671$
$x_0 = 0$	$x_1 = 0.2$	$x_2 = 0.4$

The Runge – Kutta Methods

Consider the initial value problem:

$$y' = f(x, y); \;\; y(x_0) = y_0$$

Since y is a function of x, and it can be written as $y(x)$

Then by mean value theorem,

$$y(x_i + h) = y(x_i) + hy'(x_i + \theta h)$$

Where, $0 < \theta < 1$

In our usual notation, this can be written as:

$$y_{i+1} = y_i + hf\big(x_i + \theta h\,, y(x_i + \theta h)\big)$$

Now putting $\theta = \frac{1}{2}$, we obtain

$$y_{i+1} = y_i + hf\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} f_i\right)$$

And since Euler's method with spacing $\frac{h}{2}$, this formula may be expressed as:

$$d_1 = hf(x_i, y_i)$$

$$d_2 = hf(x_i + \frac{h}{2}, y_i + \frac{d_1}{2})$$

Therefore,

$$y_{i+1} = y_i + d_2$$

This is called the second order Runge – Kutta formula.

The third order formula is:

$$d_1 = hf(x_i, y_i)$$

$$d_2 = hf(x_i + \frac{h}{2}, y_i + \frac{d_1}{2})$$

$$d_3 = hf(x_i + h, y_i + 2d_2 - d_1)$$

Therefore,

$$y_{i+1} = y_i + \frac{1}{6}(d_1 + 4d_2 + d_3)$$

The fourth order Runge – Kutta formula is given as:

$$d_1 = hf(x_i\,, y_i)$$

$$d_2 = hf(x_i + \frac{h}{2}\,, y_i + \frac{d_1}{2})$$

$$d_3 = hf\left(x_i + \frac{h}{2}\,, y_i + \frac{d_2}{2}\right)$$

$$d_4 = hf(x_i + h\,, y_i + d_3)$$

Therefore,

$$y_{i+1} = y_i + \frac{1}{6}(d_1 + 2d_2 + 3d_3 + d_4)$$

Example

Solve the initial value problem value using the Runge – Kutta second order method.

$$\frac{dy}{dx} = (1 + x^2)y \;\; ; \;\; y(0) = 1 \;\; ; \;\; x = 0(0.2)0.6$$

Solution

$$f(x, y) = (1 + x^2)y \; ; \;\; x_0 = 0, y_0 = 1, h = 0.2$$

$y_0 = 1$	$y_1 = ?$	$y_2 = ?$	$y_3 = ?$
$x_0 = 0$	$x_1 = 0.2$	$x_2 = 0.4$	$x_3 = 0.6$

To find y_1

$$d_1 = hf(x_i, y_i) = 0.2(1 + x_0^2)y_0$$

$$= 0.2(1)(1) = 0.2$$

$$d_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{d_1}{2}\right) = hf(0.1, 1.1) = 0.2(1 + 0.01)1.1$$

$$= 0.2222$$

Therefore,

$$y_1 = y_o + d_2 = 1 + 0.2222$$

$$= 1.2222$$

To find y_2:

$$d_1 = hf(x_1, y_1) = 0.2(1 + x_1^2)y_1 = 0.2(1 + 0.04)(1.2222)$$

$$= 0.2542222$$

$$d_2 = hf\left(x_1 + \frac{h}{2}, y_1 + \frac{d_1}{2}\right) = hf(0.3, 1.349333) = 0.2(1 + 0.09)(1.34933)$$

$$= 0.2941546$$

Then, $y_2 = y_1 + d_2 = 1.2222 + 0.2941546 = 1.5163768$

To find y_3

$$d_1 = hf(x_2, y_2) = 0.2(1 + x_2^2)y_2 = 0.2(1 + 0.16)(1.5163768)$$

$$= 0.3517994$$

$$d_2 = hf\left(x_2 + \frac{h}{2}, y_2 + \frac{d_1}{2}\right) = hf(0.5, 1.6922785)$$

$$= 0.4230691$$

Then, $y_3 = y_2 + d_2 = 1.5163768 + 0.4230691$

$$= 1.9394459$$

Assignment

Solve the problem given below, using the Runge – Kutta fourth order method:

$$\frac{dy}{dx} = (1 + x^2)y \ ; \ \ y(0) = 1 \ ; \ \ x = 0(0.2)0.6$$

Chapter 4

Root Finding in one-dimension

This involves searching for solutions to equations of the form: $f(x) = 0$

The various methods include:

1. Bisection Method

This is the simplest method of finding a root to an equation. Here we need two initial guesses x_a and x_b which bracket the root.

Let $F_a = f(x_a)$ and $F_b = f(x_b)$ such that $F_a F_b \leq 0$ (see fig 1)

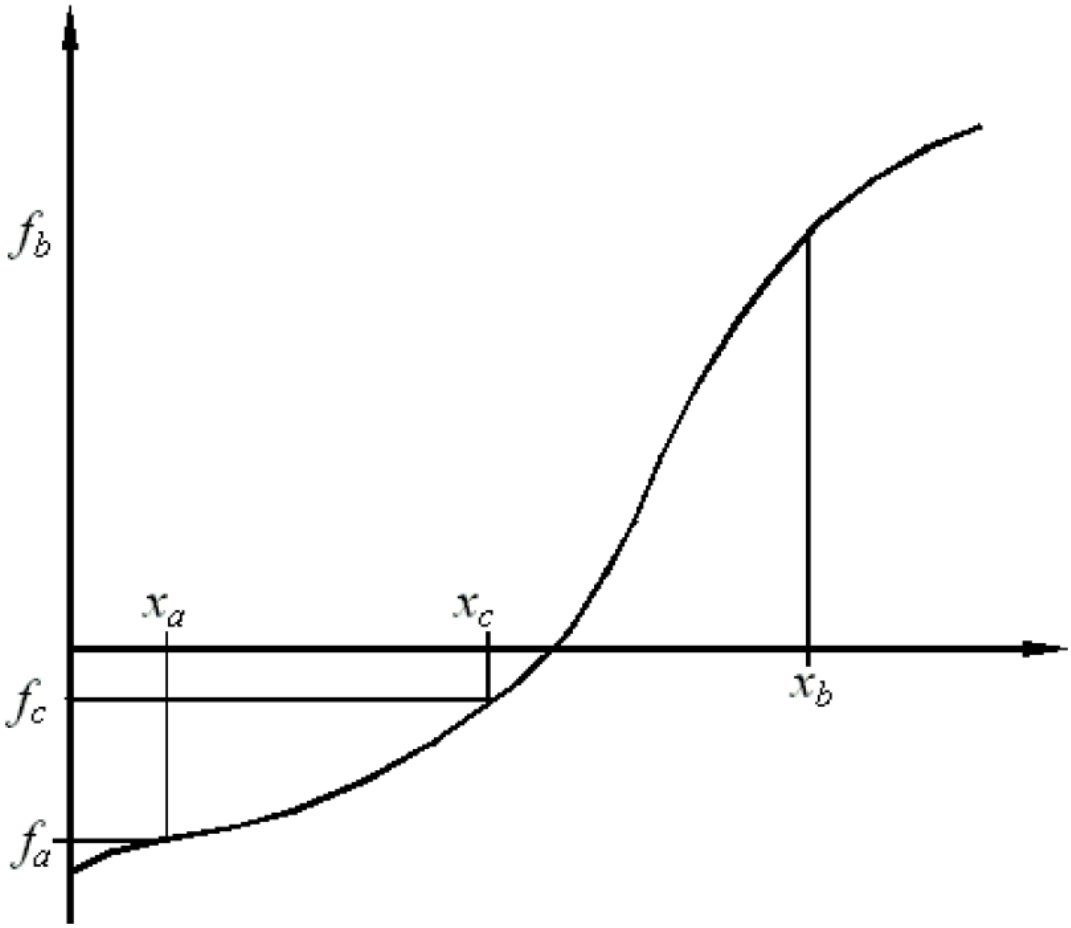


Figure 1:Graphical representation of the bisection method showing two initial guesses (x_a and x_b)

Clearly, if $F_a F_b = 0$ then one or both of x_a and x_b must be a root of $F(x) = 0$

The basic algorithm for the bisection method relies on repeated applications of:

Let $x_c = \frac{(x_a + x_b)}{2}$

If $F_c = f(c) = 0$ then, $x = x_c$ is an exact solution,

Else if $F_a F_b < 0$ then the root lies in the interval (x_a, x_c)

Else the root lies in the interval (x_c, x_b)

By replacing the interval (x_a, x_b) with either (x_a, x_c) or (x_c, x_b) (whichever brackets the root), the error in our estimation of the solution to $F(x) = 0$ is on the average, halved. ***We repeat this interval halving until either the exact root has been found or the interval is smaller than some specified tolerance.***

Hence, the root bisection is a simple but slowly convergent method for finding a solution of $F(x) = 0$, assuming the function f is continuous. It is based on the ***intermediate value theorem,*** *which states that if a continuous function f has opposite signs at some x = a and x = b(>a) that is, either $f(a) < 0$, $f(b) > 0$, or $f(a) > 0$, $f(b) < 0$ then f must be 0 somewhere on [a,b].*

We thus obtain a solution by repeated bisection of the interval and in each iteration, we pick that half which also satisfies that sign condition.

In conclusion we can consider the following steps:

Full Algorithm

1. Define F(x)
2. Read x_1 , x_2, values of x such that $F(x_1)F(x_2) < 0$
3. Read convergence term, $s = 10^{-6}$, say.
4. Calculate F(y), $y = (x_1+x_2) / 2$
5. If $abs(x_2-x_1) \leq s$, then y is a root. Go to 9
6. If $abs(x_2-x_1) > s$, Go to 7
7. If $F(x_1)F(x_2) \leq 0$, (x_1,y) contains a root, set $x_2 = y$ and return to step 4
8. If not, (y, x_2) contains a root, set $x_1 = y$ and return to step 4
9. Write the root A

Example:

Given that$F(x) = x - 2.44$, solve using the method of root bisection, the form $F(x) = 0$.

Solution:

Given that $F(x) = x - 2.44 = 0$

Therefore,

$$x - 2.44 = 0$$

Direct method gives $x = 2.44$

But by root bisection;

Let the trial value of x = *-1*

X		F(x) = x-2.44
Trial value	-1	-3.44
	0	-2.44
	1	-1.44
	2	-0.44
	3	+0.56

It is clear from the table that the solution lies between x = 2 and x = 3.

Now choosing x = 2.5, we obtain F(x) = 0.06, we thus discard x = 3 since F(x) must lie between 2.5 and 2. Bisecting 2 and 2.5, we have x = 2.25 with F(x) = -0.19.

Obviously now, the answer must lie between 2.25 and 2.5.

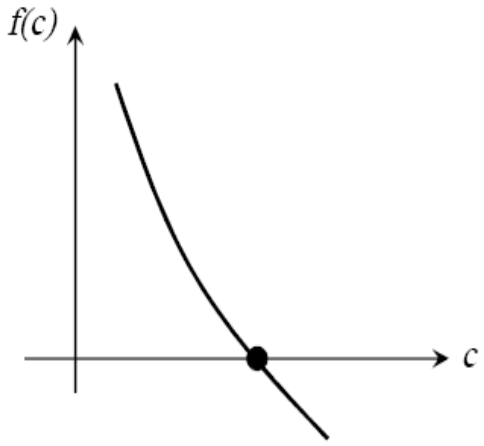
The bisection thus continues until we obtain F(x) very close to zero, with the two values of x having opposite signs.

X	F(x) = x-2.44
2.25	-0.19
2.375	-0.065
2.4375	-0.0025
2.50	0.06

When the above is implemented in a computer program, it may be instructed to stop at say, $|F(x)| \leq 10^{-4}$, since the computer may not get exactly to zero.

Example 2:

- The parachutist velocity is $v = \frac{mg}{c}(1 - e^{-\frac{c}{m}t})$
- What is the drag coefficient c needed to reach a velocity of 40 m/s if m = 68.1 kg, t = 10 s, g= 9.8 m/s^2



$$f(c) = \frac{mg}{c}(1 - e^{-\frac{c}{m}t}) - v$$

$$f(c) = \frac{667.38}{c}(1 - e^{-0.146843c}) - 40$$

1. Assume x_1 =12 and x_2=16

$f(x_1) = 6.067 \; and \; f(x_2) = -2.269$

$f(x_1)f(x_2) < 0$

2. The root: $y = (x_1 + x_2)/2 = 14$

$f(14) = 1.569$

3. $Check\ f(12).f(14) = 6.067 \times 1.569 = 9.517 > 0;$

the root lies between 14 and 16.

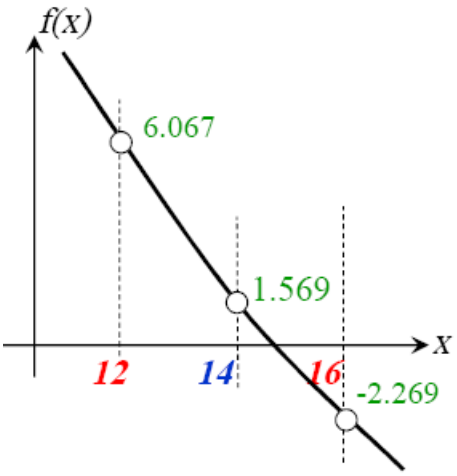
4. Set $x_1 = 14$ and $x_2 = 16$, thus the new root

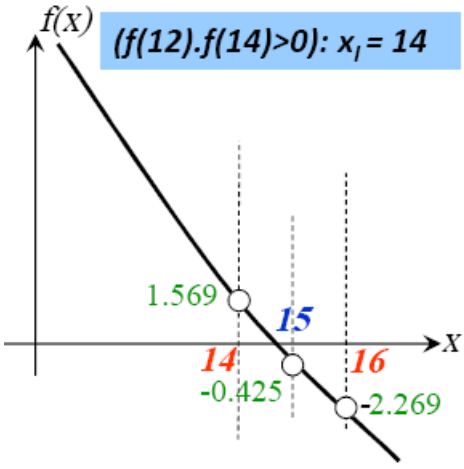
$y = (14 + 16)/2 = 15$

$f(15) = -0.425$

5. Check $f(14).f(15) = 1.569 \times -0.425 = -0.666 < 0;$

And so on





2. The Newton-Raphson Method

This is another iteration method for solving equations of the form*: F(x) = 0*, where f is assumed to have a continuous derivative f'. ***The method is commonly used because of its simplicity and great speed.*** The idea is that we approximate the graph of *f* by suitable tangents. Using an approximate value x_0 obtained from the graph of *f*, we let x be the point of intersection of the x – axis and the tangent to the curve of *f* at x_0.

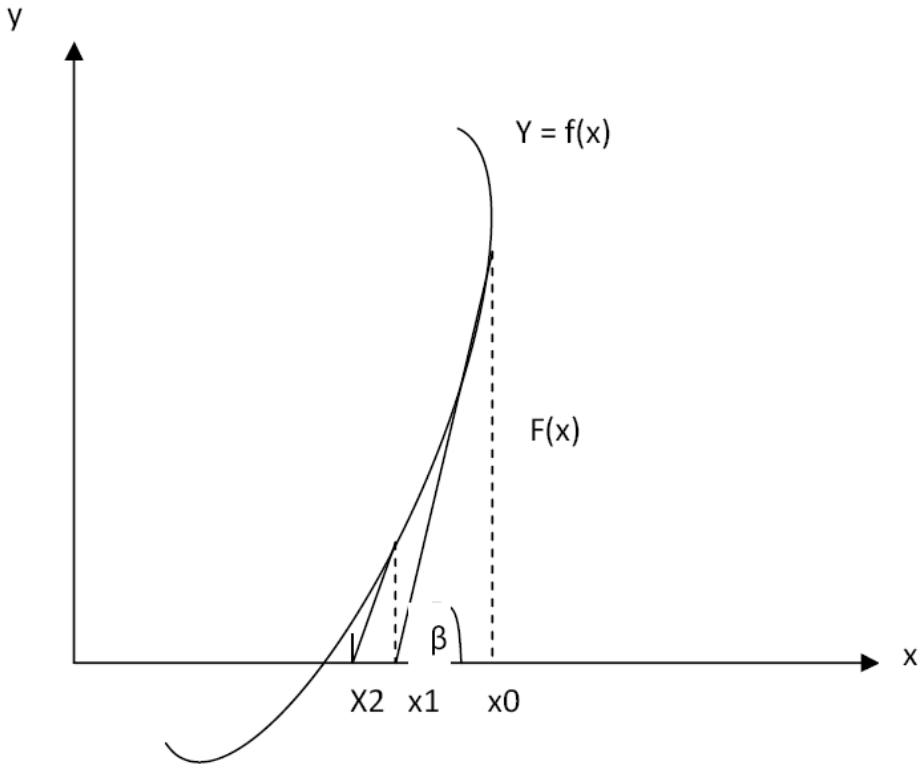


Figure 4: Illustration of the tangents to the curve in Newton-Raphson method

Then,

$\tan\beta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1}$,

Hence,

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

In the second step, we compute; $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$,

And generally,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Example:

Evaluate a real root of $x^3 + 2.1x^2 + 13.1x + 22.2 = 0$, using the Newton Raphson's method, correct to three decimal places.

Solution;

F(x) = $x^3 + 2.1x^2 + 13.1x + 22.2$

F(0) = 22.2 (positive)

Now, since all the coefficients are positive, we note that f(1), f(2), ------- are all positive. So the equation has no positive root.

We thus search in the negative side:

F (-1) = 20.2 (positive)

F (-2) = +ve = f (-3) ------ f (-11). But f (-12) is negative, so we can choose x_0 = -11.

Iteration 1

F(x) f $x^3 + 2.1x^2 + 13.1x + 22.2$

F$^{'}$ (x) = f $3x^2 + 24.2x + 13.1$

Now, with x_0 = -11

F (x_0) = F (-11) = 11.2

F$^{'}$ (x_0) = F$^{'}$(-11) = f $3(-11)^2 + 24.2(-11) + 13.1$ = 109.9

Therefore,

$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -11 - \frac{11.2}{109.9}$ = -11.1019

Iteration 2

X_1 = -11.1019

F (x_1) = f (-11.1019) = -0.2169

F$^{'}$ (x_1) = F$^{'}$ (-11.1019) = 114.1906

Therefore,.

$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ = - 11.1019 $- \frac{(-0.2169)}{114.1906}$ = - 11.100001

Iteration 3

X_2 = -11.100001

F (x_2) = F (- 11.100001) = - 0.0001131

F$'$ (x_2) = F$'$ (-11.100001) = 114.1101

Therefore,

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = -11.100001 - \frac{(-0.000113)}{114.1101} = -11.1000000$$

Now, correct to three decimal places, $x_2 = x_3$, and so, the real root is x = -11.1000.

Example 2

Set up a Newton-Raphson iteration for computing the square root *x* of a given positive number *c* and apply it to *c* = 2

Solution

We have $x = \sqrt{c}$, hence

F (x) = $x^2 - c = 0$

$$f'(x) = 2x$$

Newton-Raphson formula becomes:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{(x_k^2 - c)}{2x_k}$$

$$= \frac{2x_k^2 - x_k^2 + c}{2x_k} \quad = \frac{x_k^2 + c}{2x_k}$$

$$= {}^1\!/_2 \left(x_k + \frac{c}{x_k}\right)$$

Therefore,

For c = 2, choosing $x_0 = 1$, we obtain:

$X_1 = 1.500000$, $x_2 = 1.416667$, $x_3 = 1.414216$, $x_4 = 1.414214$, ...

Now, x_4 is exact to 6 decimal places.

Now, what happens if $f'(x_k) = 0$?

Recall, if $f(x) = 0$,and $f'(x) = 0$, we have repeated roots or multiplicity (multiple roots). The sign in this case will not change; the method hence breaks down. The method also fails for a complex solution (i.e. $x^2 + 1 = 0$)

Example :

Find the root of the equation $x^2 - 4x - 7$ near $x = 5$ to nearest thousands

Solution

We have our $x_0 = 5$. In order to use Newton's method, we also need to know the derivative of f. In this case, $f(x) = 4x - 7$, and $f'(x) = 2x - 4$.

Using Newton's method, we get the following sequence of approximations:

$$x_1 = 5 - \frac{5^2 - 4 \times 5 - 7}{2 \times 5 - 4} = 5 - \left(\frac{-2}{6}\right) = \frac{16}{3} \approx 5.33333$$

$$x_2 = \frac{16}{3} - \frac{\left(\frac{16}{3}\right)^2 - 4\left(\frac{16}{3}\right) - 7}{2\left(\frac{16}{3}\right) - 4} = \frac{16}{3} - \frac{\frac{1}{9}}{\frac{20}{3}} = \frac{16}{3} - \frac{1}{60} = \frac{319}{60} \approx 5.31667$$

$$x_3 = \frac{319}{60} - \frac{\left(\frac{319}{60}\right)^2 - 4\left(\frac{319}{60}\right) - 7}{2\left(\frac{319}{60}\right) - 4} = \frac{319}{60} - \frac{\frac{1}{3600}}{\frac{398}{60}} \approx 5.31662.$$

We can stop now, because the thousandth and ten-thousandth digits of x_2 and x_3 are the same. If we were to continu they would remain the same because we have gotten sufficiently close to the root:

$$x_4 = 5.31662 - \frac{(5.3362)^2 - 4(5.3362) - 7}{2(5.3362) - 4} = 5.31662.$$

Example

Perform four iterations of the Newton's method to find the smallest positive root of the equation $f(x) = x^3 - 5x + 1 = 0$.

Solution: We have $f(0) = 1$, $f(1) = -3$. Since, $f(0)f(1) < 0$, the smallest positive root lies in the interval $(0, 1)$. Applying the Newton's method, we obtain

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^3 - 5x_k + 1}{3x_k^2 - 5} = \frac{2x_k^3 - 1}{3x_k^2 - 5}, \qquad k = 0, 1, 2, ...$$

Let $x_0 = 0.5$. We have the following results.

$$x_1 = \frac{2x_0^3 - 1}{3x_0^2 - 5} = \frac{2(0.5)^3 - 1}{3(0.5)^2 - 5} = 0.176471,$$

$$x_2 = \frac{2x_1^3 - 1}{3x_1^2 - 5} = \frac{2(0.176471)^3 - 1}{3(0.176471)^2 - 5} = 0.201568,$$

$$x_3 = \frac{2x_2^3 - 1}{3x_2^2 - 5} = \frac{2(0.201568)^3 - 1}{3(0.201568)^2 - 5} = 0.201640,$$

$$x_4 = \frac{2x_3^3 - 1}{3x_3^2 - 5} = \frac{2(0.201640,)^3 - 1}{3(0.201640,)^2 - 5} = 0.201640.$$

Therefore, the root correct to six decimal places is $x \equiv 0.201640$.

Example

Using Newton-Raphson method solve $x \log_{10} x = 12.34$ with $x_0 = 10$.

Solution: Here $f(x) = x \log_{10} x - 12.34$. Then $f'(x) = \log_{10} x + \dfrac{1}{\log_e 10} = \log_{10} x + 0.434294$. Applying the Newton's method, we obtain

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k \log_{10} x_k - 12.34}{\log_{10} x_k + 0.434294} = \frac{(0.434294)x_k + 12.34}{\log_{10} x_k + 0.434294}, \qquad k = 0, 1, 2, ...$$

Let $x_0 = 10$. We have the following results.

$$x_1 = \frac{(0.434294)x_0 + 12.34}{\log_{10} x_0 + 0.434294} = \frac{(0.434294)(10) + 12.34}{\log_{10} 10 + 0.434294} = 11.631465.$$

$$x_2 = \frac{(0.434294)x_1 + 12.34}{\log_{10} x_1 + 0.434294} = \frac{(0.434294)(11.631465) + 12.34}{\log_{10}(11.631465) + 0.434294} = 11.594870.$$

$$x_3 = \frac{(0.434294)x_2 + 12.34}{\log_{10} x_2 + 0.434294} = \frac{(0.434294)(11.594870) + 12.34}{\log_{10}(11.594870) + 0.434294} = 11.594854.$$

We have $|x_3 - x_2| = |11.594854 - 11.594870| = 0.000016$. Therefore, We may take $x \equiv 11.594854$ as the root correct to four decimal places.