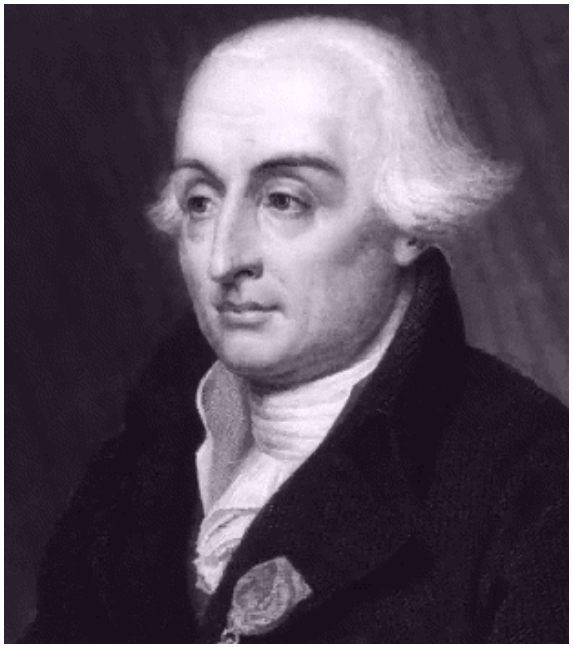

Joseph-Louis Lagrange*

Introduction to Numerical Methods

*Photo from *http:shouldersofgiants.pbworks.comwpage9628873joseph%20lagrange*

Introduction to Numerical Methods

- Some engineering problems can not be solved analytically because they are either too difficult or impossible to solve. For example, the problem of finding the solution of the following metrix.

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

  where A is the matrix of size 1000×1000.
- Such problems frequently are solved using **numerical methods** which are systematic methods that suitable for solving numerically using computational devices such calculators or computers.
- Typical numerical methods are iterative in nature and, for a well-chosen problem and good starting value, will normally converge to a correct answer.

- Many software packages are available for numerical computations such as Maple, Mathematica, Matlab, LAPACK, etc. Although, the availability of those softwares does not alleviate our effort and responsibility to *understand* the numerical methods, for example, understanding the algorithms will give you the knowledge of how to choose the software packages to suitable with your problems. Also, it will also help you to understand the quality of the software.

Solution of Equations by Iteration

One of the simple numerical method is the method of finding the solution of function $f(x) = 0$. For this problem, it is equivalent to find $x = s$ such that $f(s) = 0$. There are several well-known algorithms used for this kind of problem such as

1. Fixed-point iteration
2. Newton-Raphson's method
3. Secant method

We will consider only on the last two methods.

Solution of Equations by Iteration (Newton's method)

Algorithm : Newton's method (also known as Newton-Raphson's method)

Input : 1) given $f(x)$, 2) derivative of $f(x)$ denoted by $f'(x)$, 3) initial approximation x_0, 4) error $\varepsilon > 0$ and 5) maximum number of iteration N.

For $n = 0, 1, 2, \ldots, N-1$ **do**

1. compute $f'(x_n)$
2. If $f'(x_n) = 0$, then output *Algorithm fail* and stop (try different x_0).
3. else

   compute

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{20}$$

Solution of Equations by Iteration (Newton's method)

Algorithm : Newton's method (continue..)

4. If $|x_{n+1} - x_n| \leq \varepsilon |x_{n+1}|$, then output x_{n+1} and stop.

End

5. Output *Failure* and stop.

Solution of Equations by Iteration (Newton's method)

Example NM-1: Use Newton's method to solve for roots of $x^2 - 2$ $(x_0 = 1)$

- **n = 0**

  step 1: Compute $f'(x_0) = 2(1) = 2$

  step 2: If $f'(x_n) = 0$, then output *Algorithm fail* and stop

  $\Rightarrow$ Not true, go to step 3.

  step 3: Compute

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \Longrightarrow \qquad x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n}$$

$$= \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$$

  Thus, $x_1 = \frac{1}{2}\left(x_0 + \frac{2}{x_0}\right) = \frac{1}{2}\left(1 + \frac{2}{1}\right) = 1.5$

  step 4: If $|x_{n+1} - x_n| \leq \varepsilon |x_{n+1}|$, then output x_{n+1} and stop.

  $\Rightarrow$ Suppose it's Not. Hence, do the next iteration.

Solution of Equations by Iteration (Newton's method)

- **n = 1**

  step 1: $f'(x_1) = 2(1.5) = 3$

  step 2: No !

  step 3: $x_2 = \frac{1}{2}\left(1.5 + \frac{2}{1.5}\right) = 1.41667$

  step 4: No !

- **n = 2**

  step 1: $f'(x_2) = 2(1.41667) = 2.83334$

  step 2: No !

  step 3: $x_3 = \frac{1}{2}\left(1.41667 + \frac{2}{1.41667}\right) = 1.41422$

  step 4: No !

$$\begin{array}{ccc} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{array}$$

Solution of Equations by Iteration (Newton's method)

From the previous example, it can be seen that it is more convenient to use table in the computation as shown in the next example.

Example NM-2: Find the positive solution of $2\sin(x) = x$ $(x_0 = 2)$

Form $f(x) = 0 \Rightarrow f(x) = x - 2\sin(x) = 0 \Rightarrow f'(x_n) = 1 - 2\cos(x_n)$

$$x_{n+1} = x_n - \frac{x_n - 2\sin(x_n)}{1 - 2\cos(x_n)} = \frac{2(\sin(x_n) - x_n\cos(x_n))}{1 - 2\cos(x_n)} = \frac{N_n}{D_n}$$

n	x_n	N_n	D_n	x_{n+1}
0	2.00000	3.48318	1.83229	1.90100
1	1.90100	3.12470	1.64847	1.89552
2	1.89552	3.10500	1.63809	1.89550
3	1.89550	3.10493	1.63806	1.89549

*** The exact solution is $\mathbf{x = 1.895494}$ ***

Solution of Equations by Iteration (Newton's method)

Example NM-3: Find the root of $f(x) = x^3 + x - 1$ $(x_0 = 1)$

$$x_{n+1} = x_n - \frac{x_n^3 + x_n - 1}{3x_n^2 + 1} = \frac{2x_n^3 = 1}{3x_n^2 + 1} = \frac{N_n}{D_n}$$

n	x_n	N_n	D_n	x_{n+1}
0	1.00000	3.00000	4.00000	0.75000
1	0.75000	1.84375	2.68750	0.68605
2	0.68605	1.64580	2.41199	0.68234
3	0.68234	1.63538	2.39676	0.68233

Solution of Equations by Iteration (Secant method)

- Sometime, $f'(x)$ is difficult to express or find.
- We replace the derivative of $f(x)$ with the following approximation.

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{21}$$

- Hence, in Secant method, the following formula is used instead of (20).

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{22}$$

  Observe that we need not to compute the derivative of $f(x)$ in (22).
- The remaining steps of Secant method are same as Newton's method.

Solution of Equations by Iteration (Secant method)

The following figure shows how Newton's method and Secant method approach the correct solution in each iteration.

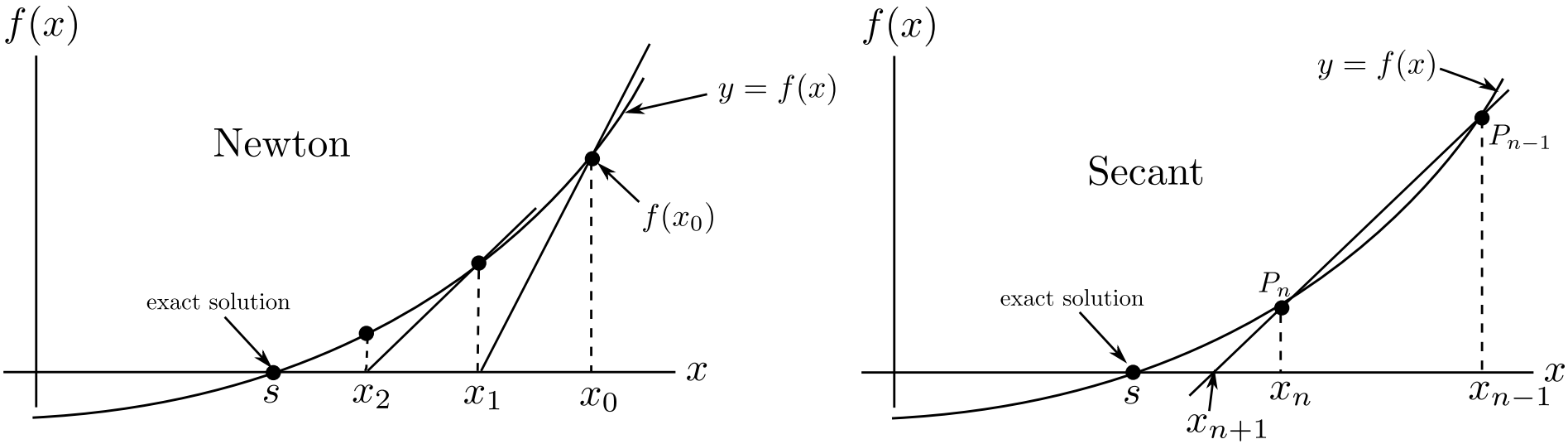

Figure 1: Graphical illustration how Newton's and Secant method approach to solution.

Interpolation

- We are given the values of a function $f(x)$ at different points $x_0, x_1, \ldots, x_n$.

x	x_0	x_1	x_2	$\cdots\cdots$	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	$f(x_2)$	$\cdots\cdots$	$f(x_n)$

Then, we want to approximate value of function $f(x)$ for **new x** that lies between these points for which the function values are given.

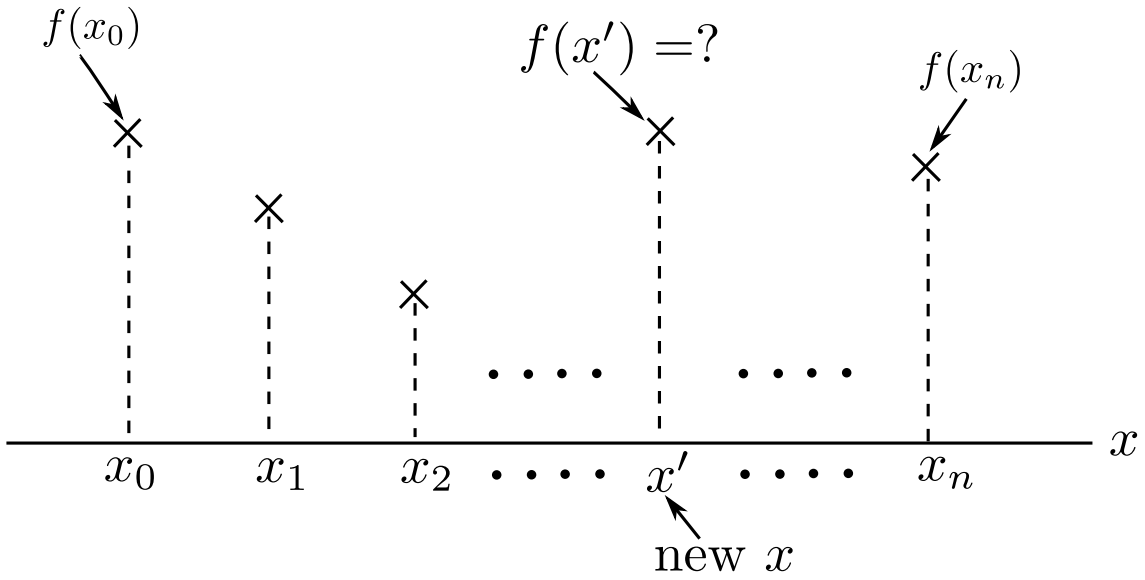

Figure 2: Corresponding plot of $f(x)$.

Interpolation

- Where do these given functions come from? $\Rightarrow$ they may come from *mathematical functions* or *empirical results*.
- For the subsequent explanation, let's denote as follow

$$\begin{aligned} \text{Denote :} \quad & f_0 = f(x_0), \quad f_0 = f(x_0), \quad \cdots \quad , f_n = f(x_n) \\ \text{In order pairs:} \quad & (x_0, f_0), \quad (x_1, f_1), \quad \cdots \quad , (x_n, f_n) \end{aligned}$$

- Standard idea in interpolation is to find a polynomial $p_n(x)$ of degree n (or less) such that

$$p_n(x_0) = f_0, \quad p_n(x_1) = f_1, \quad \cdots \quad , \quad p_n(x_n) = f_n.$$

Interpolation (Lagrange Interpolation)

- How do we find polynomial $p_n(x)$?
  $\Rightarrow$ Lagrange interpolation
  $\Rightarrow$ Newton's method
  $\Rightarrow$ Spline interpolation

  We will only introduce Lagrange interpolation in this section.

Lagrange Interpolation

Given $(x_0,f_0),(x_1,f_1),\ldots,(x_n,f_n)$, the Lagrange interpolation polynomial is defined by

$$p_n(x) = L_0(x)\cdot f_0 + L_1(x)\cdot f_1 + \ldots\ldots + L_n(x)\cdot f_n, \tag{23}$$

where $L_j(x)$ is the polynomial such that 1) $L_j(x_j) = 1$ and 2) $L_j(x_i) = 0$, for $i \neq j$, for $j = 0,\ldots,n$.

Interpolation (Lagrange Interpolation)

a) $n = 1$ (Linear interpolation)

$$p_1(x) = \underbrace{\left(\frac{x - x_1}{x_0 - x_1}\right)}_{L_0(x)} f_0 + \underbrace{\left(\frac{x - x_0}{x_1 - x_0}\right)}_{L_1(x)} f_1, \tag{24}$$

Observe that 1) we need two initial conditions (x_0, f_0) and (x_1, f_1), 2) $L_0(x) = 1$ at x_0 and 0 at x_1, while, $L_1(x)$ is the opposite.

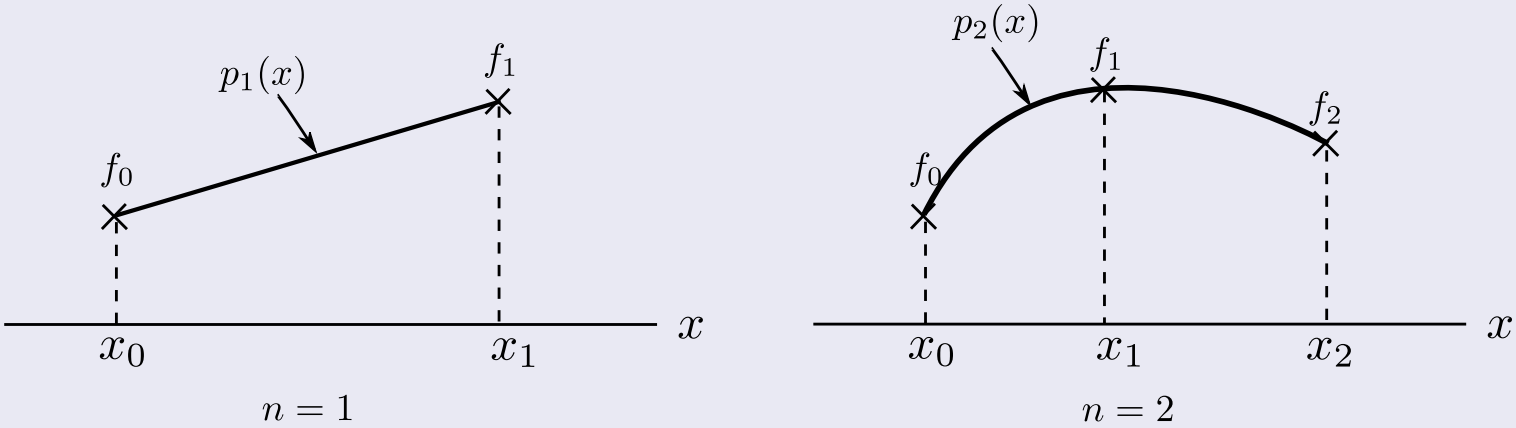

Figure 3: How Lagrange polynomial is created for order $n = 1$ (linear) and $n = 2$ (quadratic)

Interpolation (Lagrange Interpolation)

b) $n = 2$ (Quadratic interpolation)

$$p_2(x) = L_0(x) \cdot f_0 + L_1(x) \cdot f_1 + L_2(x) \cdot f_2, \tag{25}$$

where

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \tag{26}$$

$$L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \tag{27}$$

$$L_2(x) = \frac{(x - x_2)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \tag{28}$$

For this case, we require three initial conditions: (x_0, f_0), (x_1, f_1), and (x_2, f_2)

Interpolation (Lagrange Interpolation)

In general case

$$p_n(x) = L_0(x) \cdot f_0 + L_1(x) \cdot f_1 + \ldots\ldots + L_n(x) \cdot f_n, \tag{29}$$

where

$$L_j(x) = \frac{l_j(x)}{l_j(x_j)} \tag{30}$$

$$l_j(x) = \begin{cases} (x-x_0)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n) & : 0 < j < n; \\ (x-x_0)(x-x_1)\cdots(x-x_{n-1}) & : j = n; \\ (x-x_1)(x-x_2)\cdots(x-x_0) & : j = 0. \end{cases} \tag{31}$$

and we need $n+1$ initial conditions: $(x_0, f_0), \ldots, (x_n, f_n)$

Example NM-4: Suppose we know that $\ln(9.0) = 2.1972$ and $\ln(9.5) = 2.2513$. Find an approximate value of $\ln(9.2)$.

From a given data, we have

$$x_0 = 9.0, x_1 = 9.5, f_0 = 2.1972, f_1 = 2.2513.$$

Then,

$$L_0(x) = \frac{x-9.5}{9.0-9.5} = -2(x-9.5)$$
$$L_1(x) = \frac{x-9.0}{9.5-9.0} = 2(x-9.0)$$

Finally, we get

$$\therefore p_1(x) = -2(x-9.5)(2.1972) + 2(x-9.0)(2.2513).$$

For $x = 9.2$, the linear $(n = 1)$ lagrange approximation, $p_1(x)$, of $\ln(9.2)$ is

$$\begin{aligned} p_1(x = 9.2) &= -2(9.2 - 9.5)(2.1972) + 2(9.2 - 9.0)(2.2513) \\ &= 2.2188 \quad \text{(exact value = 2.2192)} \quad \blacksquare \end{aligned}$$

Example NM-5: Given $\ln(9.0) = 2.1972$, $\ln(9.5) = 2.2513$, and $\ln(11.0) = 2.3979$. Find an approximate value of $\ln(9.2)$.

From a given data.

$$x_0 = 9.0,\ x_1 = 9.5,\ x_2 = 11.0$$

$$f_0 = 2.1972, f_1 = 2.2513, f_2 = 2.3979$$

Interpolation (Lagrange Interpolation)

Compute $L_0(x)$, $L_1(x)$, and $L_2(x)$.

$$L_0(x) = \frac{(x-9.5)(x-11.0)}{(9.0-9.5)(9.0-11.0)} = x^2 - 20.5x + 104$$

$$L_1(x) = \frac{(x-9.0)(x-11.0)}{(9.5-9.0)(9.5-11.0)} = \frac{4}{3}(x^2 - 20x + 99)$$

$$L_2(x) = \frac{(x-9.0)(x-9.5)}{(11.0-9.0)(11.0-9.5)} = \frac{1}{3}(x^2 - 18.5x + 85.5)$$

Hence, we obtain $p_2(x)$.

$$\begin{aligned} p_2(x) = {} & (x^2 - 20.5x + 104)(2.1972) \\ & + \frac{4}{3}(x^2 - 20x + 99)(2.2513) \\ & + \frac{1}{3}(x^2 - 18.5x + 85.5)(2.3979). \end{aligned}$$

Interpolation (Lagrange Interpolation)

For $x = 9.2$, the quadratic $(n = 2)$ lagrange approximation, $p_2(x)$, of $\ln(9.2)$ is

$$p_2(x = 9.2) = 2.2192 \quad \text{(exact value = 2.2192)} \quad \blacksquare.$$

Numerical Integration

- Definite integral is defined as

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

  which is equal to the area under the curve $f(x)$.
- In practical, we will deal with $f(x)$ that is too complicated to compute analytically. So, in order to find the value of the definite integral (or J), it has to be evaluated numerically. There are several well-known numerical integral algorithms available in both free and commercial software packages. However, for introduction, we will only discuss three basic algorithms 1) Rectangular rule, 2) Trapezoidal rule, and 3) Simpson's rule.

Rectangular rule

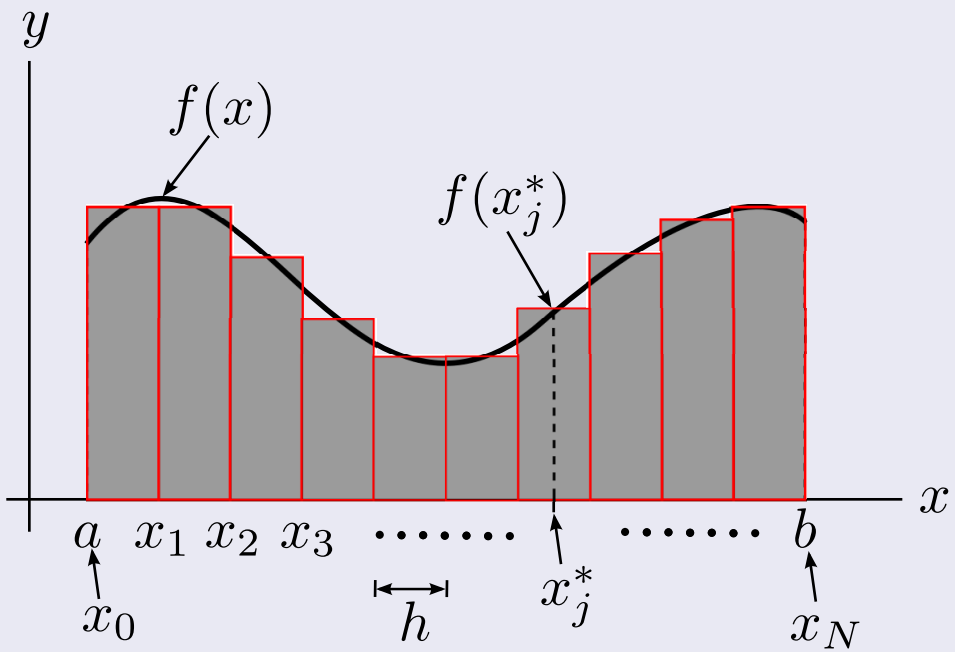

Figure 4: Graphical illustration of how integration is approximated using rectangular rule.

Rectangular rule is the simplest rule to compute the definite integral. In this method, the interval $[a, b]$ is divided into N equally subintervals, each of length $h = (b - a)/n$. Then, the value I is approximately equal to the summation of

Numerical Integration

$$J \approx R(x_1^*) + R(x_2^*) + \cdots + R(x_N^*),$$

where $R(x_j^*) = h \cdot f(x_j^*)$ is the area of rectangular j and $x_j^* = (x_{j-1} + x_j)/2$ is the midpoint of interval j, for $j = 1, \ldots, N$. Hence, for rectangular rule

$$J = \int_a^b f(x)dx \approx h\left[f(x_1^*) + f(x_2^*) + \cdots + f(x_N^*)\right] \tag{32}$$

It can be seen that the summation of all rectangular areas is not *exactly* equal to the area under the curve $f(x)$. In other word, there are some losses the due to approximating the value of J using rectangular rule. A better approximation is trapezoidal and Simpson's rule.

Trapezoidal rule

Instead of using rectangular shape in approximating the area under the curve $f(x)$, **trapezoidal rule** uses trapezoid area as shown in the following figure.

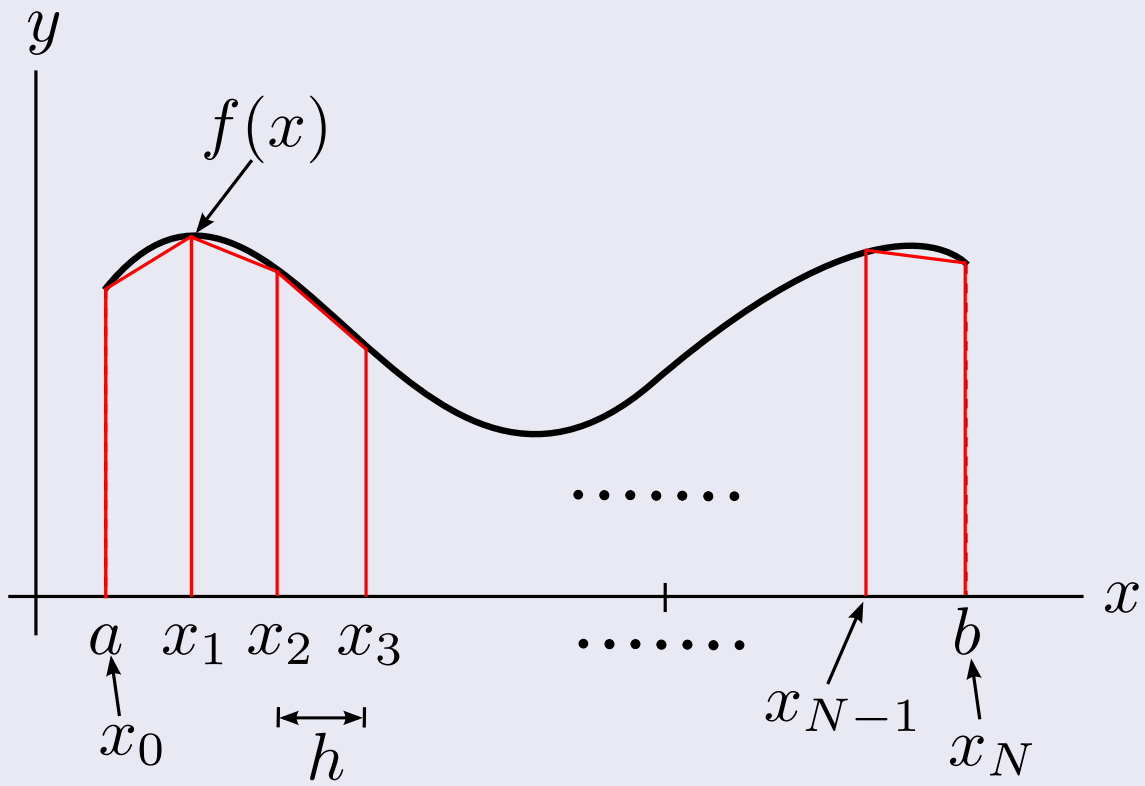

Figure 5: Graphical illustration of how integration is approximated using trapezoidal rule

Numerical Integration

The interval $[a,b]$ is also subdivided into N equally subintervals of length h. For subinterval j, the corresponding trapezoid area is equal to $\frac{1}{2}[f(x_{j-1})+f(x_j)]h$. Hence, for trapezoidal rule

$$J=\int_a^b f(x)dx \approx \overbrace{\left\{\frac{1}{2}[f(a)+f(x_1)]h\right\}}^{\text{trapezoid 1}}+\cdots+\overbrace{\left\{\frac{1}{2}[f(x_{N-1}+f(b)]h\right\}}^{\text{trapezoid N}}$$

$$\approx \left[\frac{1}{2}f(a)+f(x_1)+f(x_2)+\cdots+f(x_{N-1})+\frac{1}{2}f(b)\right]h \qquad (33)$$

Although, trapezoidal rule provides a better approximation of J, it is easily seen that there are still some losses using this rule. A more better method is Simpson's rule which uses the quadratic curve instead of straight line to approximate the area under curve $f(x)$ for each interval. See [6] for more detail including other more sophisicated methods.

Example NM-6: Evaulate $J = \int_0^1 e^{-x^2} dx$ with $N = 10$

We know that $h = (1-0)/10 = 0.1$. The following tables show how to approximate the value of J using rectangular and trapezoidal rule.

Numerical Integration

Rectangular rule

j	x_j	x_j^*	$f(x_j^*) = e^{-(x_j^*)^2}$
0	0.0		
1	0.1	0.05	0.99750
2	0.2	0.15	0.97775
3	0.3	0.25	0.93941
4	0.4	0.35	0.88471
5	0.5	0.45	0.81669
6	0.6	0.55	0.73897
7	0.7	0.65	0.65541
8	0.8	0.75	0.56978
9	0.9	0.85	0.48554
10	1.0	0.95	0.40555
Sums			7.47131

$$J \approx (7.47131)(0.1) = 0.74713 \qquad \blacksquare$$

Trapezoidal rule

j	x_j	x_j^2	$f(x_j) = e^{-x_j^2}$	
0	0.0	0.00	1.00000	
1	0.1	0.01		0.99005
2	0.2	0.04		0.96079
3	0.3	0.09		0.91393
4	0.4	0.16		0.85214
5	0.5	0.25		0.77880
6	0.6	0.36		0.69768
7	0.7	0.49		0.61263
8	0.8	0.64		0.52729
9	0.9	0.81		0.44486
10	1.0	1.00	0.36788	
Sums			1.36788	6.77817

$$J \approx \left[\frac{1}{2}(1.36788) + 6.77817\right](0.1) = 0.74621 \qquad \blacksquare$$

Bit error rate of binary phase-shift keying (BPSK) system [7]

Basic communication model is shown in the following figure

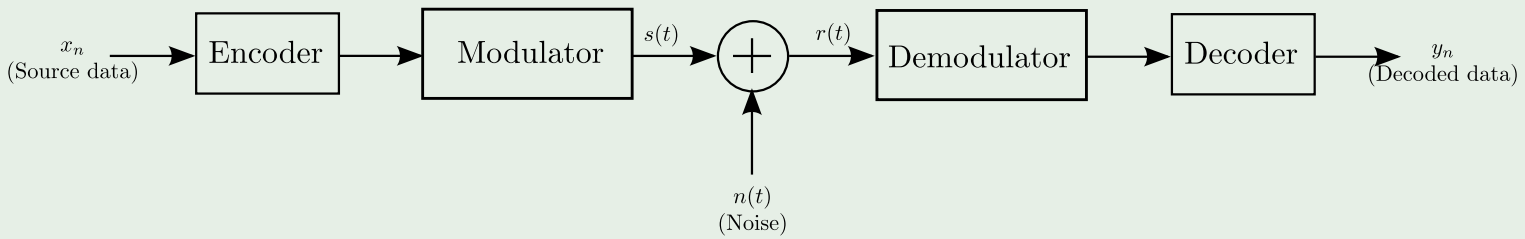

Figure 6: Basic communication models.

Suppose that the source x_n is binary data. In the testing of BPSK transmission system, we try to send the data through the additive white Gaussian noise (AWGN) channel at various signal-to-noise ratios (SNR)[a]. At the receiver, the error rates are calculated by

$$\frac{\#\text{ mismatch between source data } x_n \text{ and decoded data } y_n}{\#\text{ total source data}}.$$

[a] SNR is the ratio between the average power of modulated signal $s(t)$ to the average power of noise in the channel. Usually, it is considered in decibel (dB)

Applications: Interpolation

The results of approximated error rate are shown in the table below.

SNR (in dB)	0.00	2.00	4.00	8.00
Error rate	0.50000	0.02275	0.00234	0.00003

Table-1: SNR vs Error Rate

Using the above data, let's approximate the performance of BPSK system at SNR = 6.00 dB. using $p_3(x)$.

From the table, we can compute $L_0(x), \ldots, L_3(x)$ as follow.

$$L_0(x) = \frac{(x-2.00)(x-4.00)(x-8.00)}{(0.00-2.00)(0.00-4.00)(0.00-8.00)}$$
$$= -\frac{1}{64}x^3 + \frac{7}{32}x^2 - \frac{7}{8}x + 1$$

Applications: Interpolation

$$
\begin{aligned}
L_1(x) &= \frac{(x-0.00)(x-4.00)(x-8.00)}{(2.00-0.00)(2.00-4.00)(2.00-8.00)} \\
&= \frac{1}{24}x^3 - \frac{1}{2}x^2 + \frac{4}{3}x \\
L_2(x) &= \frac{(x-0.00)(x-2.00)(x-8.00)}{(4.00-0.00)(4.00-2.00)(4.00-8.00)} \\
&= -\frac{1}{32}x^3 + \frac{5}{16}x^2 - \frac{1}{2}x \\
L_3(x) &= \frac{(x-0.00)(x-2.00)(x-4.00)}{(8.00-0.00)(8.00-2.00)(8.00-4.00)} \\
&= \frac{1}{192}x^3 - \frac{1}{32}x^2 + \frac{1}{24}x
\end{aligned}
$$

Applications: Interpolation

So, now we have $p_3(x)$.

$$\begin{aligned} p_3(x) &= L_0(x)(0.5) + L_1(x)(0.02275) + L_2(x)(0.00234) + L_3(x)(0.00003) \\ &= 0.00694x^3 + 0.09873x^2 + 0.40834x + \frac{1}{2} \end{aligned}$$

Hence, the interpolated error rate at SNR = 6 dB is

$$\begin{aligned} p_3(x=6) &= 0.00694(6)^3 + 0.09873(6)^2 + 0.40834(6) + \frac{1}{2} \\ &= 0.1058 \quad \blacksquare \end{aligned}$$

Observe that the interpolated error rate at SNR 6 dB is NONSENSE since it is more than those of SNR 2 and 4 dB. In fact, the more SNR is used, the lower error rate is achieved. This unusual result causes by the characteristic of lagrange polynomial. The better interpolated results can just be obtained using $p_1(x)$ or $p_2(x)$.

Applications: Interpolation

In general, the performance of the communication system is measured in the sense of *average probability of bit error* or *bit error rate (BER)*. When the binary PSK modulation is used to transmit the source data through additive white Gaussian noise channel (AWGN), bit error rate of BPSK signaling is

$$BER = \frac{1}{2} erfc(\sqrt{SNR}), \tag{34}$$

where $erfc(u)$ is called *complementary error function* and is defined by

$$erfc(u) = \frac{2}{\sqrt{\pi}} \int_u^{\infty} e^{-t^2} dt \tag{35}$$

Recall the previous example, when SNR = 6 dB, the correct error rate should be equal to $\frac{1}{2} erfc\sqrt{6} = 0.00027$.

Summary

- In practical, most of engineering problems can not be solved analytically due to the complexity of the problems. Instead, they are solved numerically using some systematic methods that can be implemented in computational devices, i.e, calculators, computers, etc. Mostly such numerical methods are iterative with well-chosen starting value. At present, many software packages are available to be used such as Maple, Matlab, or Mathematica.
- Although, many numerical algorithms are implemented already in software packages and are easily to be used, the understanding of such algorithms is still necessary since it gives the knowledge of how to choose the suitable software packages for the interesting problems.
- Newton-Raphson's and Secant methods are basic algorithms to find the root of functions. In order to apply Newton-Raphson method, the derivative of the interesting function is necessarily required, while secant method has no such requirement.

Summary

- Interpolation is the numerical method that is used to approximate (unknown) value of the function for new interesting data within the range of known data set. The standard idea of interpolation is to find a polynomial such that all known data sets are defined on such polynomial. Generally, Lagrange and Spline polynomial are used.
- It is obvious that in most of practical engineering problems we will face of too complicated integral function that might not be computed analytically. Hence, many numerical integration algorithms are essentially provided in commercial software packages. Basic numerical integration algorithms are rectangular, tracpezoidal and Simpson's methods.

Exercise (Introduction to Numerical Methods)

In Exercise 1-3, find the root of $f(x)$ using Newton's and Secant methods for a given ε and initial point x_0. Then, compare number of iterations to approach the appropriate solution.

Exer 1. $f(x) = x^3 - x^2 - 2$ with $x_0 = 1.0$ and $\varepsilon = 0.005$.
(**Answer:** 1.69562.)

Exer 2. $f(x) = x^4 - x + 0.2$ with $x_0 = 1.0$ and $\varepsilon = 0.001$.
(**Answer:** 0.92169.)

Exer 3. $f(x) = x + \ln(x) - 2$ with $x_0 = 2.0$ and $\varepsilon = 0.001$.
(**Answer:** 1.55715.)

Exercise (Introduction to Numerical Methods)

In Exercise 4-6, apply Lagrange interpolation to approximate $f(x = c)$ for a given order n.

Exer 4. $f(x) = \int_0^\infty t^{x-1} e^{-t} dt = \Gamma(x)$, called *Gamma function*. Approximate $f(1.005)$ and $f(1.015)$ using p_2 with $f(1.00) = 1.0000$, $f(1.02) = 0.9888$, and $f(1.04) = 0.9784$. (**Answer:** 0.99713 and 0.99156, respectively.)

Exer 5. $f(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-w^2} dw = \text{erf}(x)$, called *error function*. Approximate $f(0.75)$ using p_2 with $f(0.25) = 0.27633$, $f(0.5) = 0.52050$, and $f(1.0) = 0.84270$. (**Answer:** 0.71116.)

Exer 6. $f(x) = \int_0^x \frac{\sin(t)}{t} dt = \text{Si}(x)$, called *Sine Integral*. $\text{Si}(x)$. Approximate $f(0.5)$ and $f(1.5)$ using p_2 with $f(0.0) = 0$, $f(1.0) = 0.9461$, and $f(2.0) = 1.6054$. (**Answer:** 0.4931 and 1.3247, respectively.)

Exercise (Introduction to Numerical Methods)

In Exercise 7-9, Numerically calculate integral as indicated using Rectangular and Trapezoidal rule. Then, compare the results from both methods.

Exer 7. $f(x) = \int_1^x \frac{1}{t} dt$. Evaluate $f(2)$ with $n = 10$.

Exer 8. $f(x) = \int_0^x te^{-t} dt$. Evaluate $f(4)$ with $n = 10$.

Exer 9. $f(x) = \int_0^x \frac{1}{\cos^2(t)} dt$. Evaluate $f(1)$ with $n = 10$.