

Numerical Analysis

DWE3214



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023



Brook Taylor

1685 - 1731



Colin Maclaurin

1698 - 1746

Unit-0: Introduction

Syllabus

Introduction to Numerical Analysis

Part-I: Basic Tools

Unit-1: Error Analysis

- Measuring Errors
- Sources of Error
- Consistency, Order, Smoothness and Convergence

Unit-2: Roots of equations (Nonlinear Equations)

- Bisection Method
- False-Position Method (Optional)
- Newton-Raphson Method
- Secant Method (Optional)

Unit-3: Simultaneous Linear algebraic Equations

- Direct Methods
 - Review of Determinants and Matrices
 - Cramer's Rule
 - Gauss-Elimination method (simple and partial pivoting methods)
 - Gauss-Jordan Method
 - Matrix Inversion method
- Indirect (Iterative) Method
 - Jacobi Method
 - Gauss-Seidel Method
 - Successive Over-Relaxation Method

Unit-4: Numerical Differentiation and Integration

- Numerical differentiation using difference method
- Numerical Integration, Trapezoid and Simpson's Rules
- Extrapolation of Errors

Unit-5: Interpolation and Curve Fitting

- Direct Fit Polynomial
- Least Squares Method
- Logarithmic regression (Optional)
- Exponential regression (Optional)

- Linear interpolation , Quadratic Interpolation
- Lagrange Interpolation (Optional)
- Newton Divided Difference Interpolation (Optional)

Part-II: Numerical Solutions of Ordinary Differential Equations

Unit-6: Initial Value Problem

- Euler's Method
- Runge-Kutta 2nd
- Runge-Kutta 4th
- Higher Order Equations

Unit-7: Boundary Value Problem

- Equilibrium (Finite Difference) Method

Part-III: Numerical Solutions of Partial Differential Equations

Unit-8: PDEs

- Elliptic Equations
- Parabolic Equations
- Hi-parabolic Equations
- Advanced Application (Case Studies based on each department interests).

References:

- Numerical Methods for Engineers, S. C. Chapra and R. P Canale, McGraw-Hill, 6th edition 2010.
- Numerical Methods for Engineers and Scientists by Joe D. Hoffman, 2nd Edition.
- Lectures on Numerical Analysis by Dennis Deturck and Herbert S. Wilf.
- Numerical Analysis Using MATLAB® and Excel® by Steven T. Karris, 3rd Edition.
- Numerical Methods in Engineering with MATLAB® by Jaan Kiusalaas.
- Engineering Analysis- Interactive Methods and Programs with FORTRAN, QuickBasic, MATLAB, and Mathematica by Y. C. Pao.

د. حسن مجيد حسون الدلفي ومحمود عطا الله مشكور.-التحليل الهندسي والعددي التطبيقي -

Assessment method:

30% course exam, 10% homework and self-initiative, 10% Lab, and 50% final exam.

Introduction:

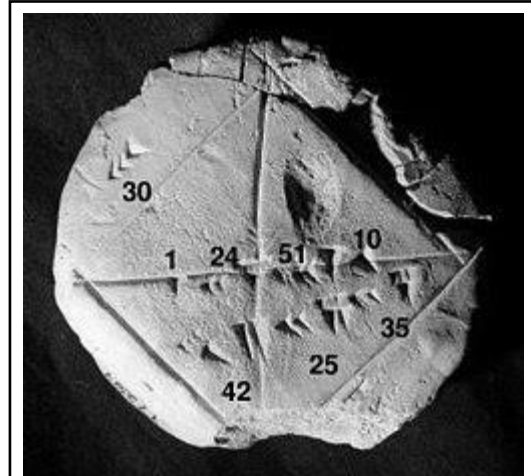
Numerical calculations obviously involve the manipulation (i.e., addition, multiplication, etc.) of numbers. Numbers can be integers (e.g., 4, 17, -23, etc.) fractions (e.g., $1/2$, $-2/3$, etc.), or an infinite string of digits (e.g., $\Pi=3.1415926535\dots$).

Wikipedia: Numerical Analysis is the study of algorithms that use numerical approximation (as opposed to symbolic manipulations) for the problems of mathematical analysis. It is the study of numerical methods that attempt at finding approximate solutions of problems rather than the exact ones. Numerical analysis finds application in all fields of engineering and the physical sciences, and in the 21st century also the life and social sciences, medicine, business and even the arts. Current growth in computing power has enabled the use of more complex numerical analysis, providing detailed and realistic mathematical models in science and engineering. Examples of numerical analysis include: ordinary differential equations as found in celestial mechanics (predicting the motions of planets, stars and galaxies), numerical linear algebra in data analysis and stochastic differential equations and Markov chains for simulating living cells in medicine and biology.

Before modern computers, numerical methods often relied on hand interpolation formulas, using data from large printed tables. Since the mid-20th century, computers calculate the required functions instead, but many of the same formulas continue to be used in software algorithms.

The numerical point of view goes back to the earliest mathematical writings. A tablet from the Yale Babylonian Collection (YBC 7289), gives a sexagesimal numerical approximation of the square root of 2, the length of the diagonal in a unit square.

Numerical analysis continues this long tradition: rather than giving exact symbolic answers translated into digits and applicable only to real-world measurements, approximate solutions within specified error bounds are used.



Babylonian clay tablet YBC 7289 (c. 1800–1600 BC) with annotations. The approximation of the square root of 2 is four sexagesimal figures, which is about six decimal figures. $1 + 24/60 + 51/60^2 + 10/60^3 = 1.41421296\dots$

Example:

A parachutist of mass 68.1 kg jumps out of a stationary hot air balloon. Compute velocity prior to opening the chute. The drag coefficient is equal to 12.5 kg/s.

Analytical Solution:

$$F = ma \Rightarrow a = \frac{F}{m}$$

Where:

F = net force acting on the body (N, or kg m/s²),

m = mass of the object (kg),

a = acceleration (m/s²).

$$\frac{dv}{dt} = \frac{F}{m}$$

$$F = F_D + F_U$$

$$F_D = mg \quad (g=9.81 \text{ m/s}^2)$$

$$F_U = -cv \quad (c=\text{dragging coefficient kg/s})$$

$$\frac{dv}{dt} = \frac{mg-cv}{m}$$

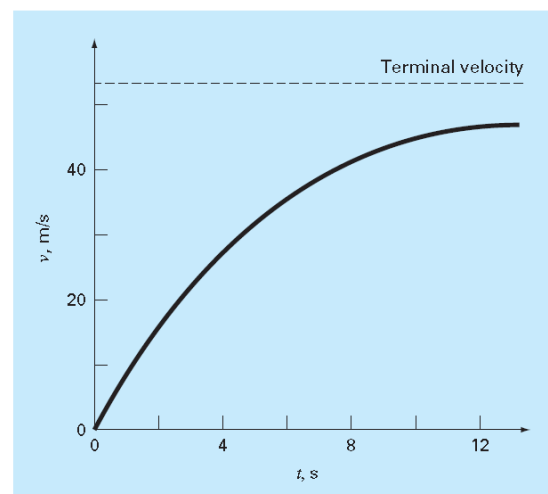
$\frac{dv}{dt} = g - \frac{c}{m}v$, this is a differential equation having the analytical solution:

$$v(t) = \frac{gm}{c} [1 - e^{-(\frac{c}{m})t}]$$

$$v(t) = \frac{9.81(68.1)}{12.5} \left[1 - e^{-(\frac{12.5}{68.1})t} \right] = 53.44(1 - e^{-0.18355t})$$



t, s	$v, m/s$
0	0.00
2	16.42
4	27.80
6	35.68
8	41.14
10	44.92
12	47.54
∞	53.44



Numerical Solution:

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

$$\frac{dv}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\Delta v}{\Delta t}$$

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m} v(t_i)$$

This equation can be rearranged to be:

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m} v(t_i) \right] (t_{i+1} - t_i)$$

New value = old value + slope × step size

This approach is formally called Euler's method and will be discussed thoroughly in Part-II.

Hence:

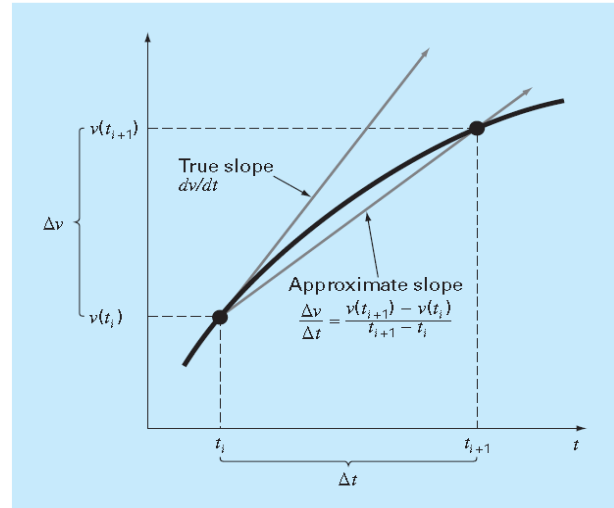
1: $t = 0$ to 2 s

$$v = 0 + \left[9.81 - \frac{12.5}{68.1} (0) \right] 2 = 19.62 \text{ m/s}$$

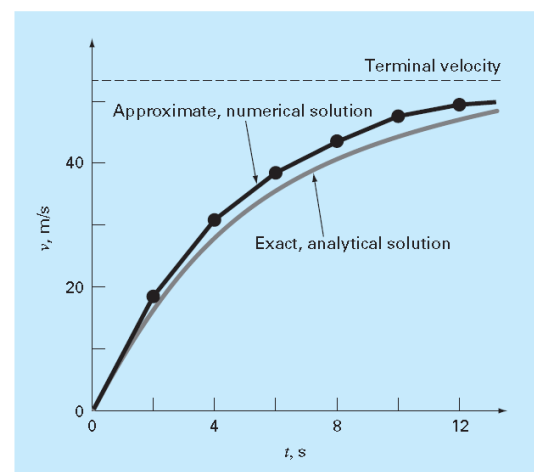
2: $t = 2$ to 4 s

$$v = 19.62 + \left[9.81 - \frac{12.5}{68.1} (19.62) \right] 2 = 32.04 \text{ m/s}$$

And so on...



$t, \text{ s}$	$v, \text{ m/s}$
0	0.00
2	19.62
4	32.04
6	39.90
8	44.87
10	48.02
12	50.01
$\infty \cong 1000000$	53.44



Numerical Analysis

DWE3214

PART-I: BASIC TOOLS

Unit-1: Error Analysis



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-1: Error Analysis

Introduction:

Numerical calculations obviously involve the manipulation (i.e., addition, multiplication, etc.) of numbers. Numbers can be integers (e.g., 4, 17, -23, etc.) fractions (e.g., $1/2$, $-2/3$, etc.), or an infinite string of digits (e.g., $\pi=3.1415926535\dots$). When dealing with numerical values and numerical calculations, there are several concepts that must be considered:

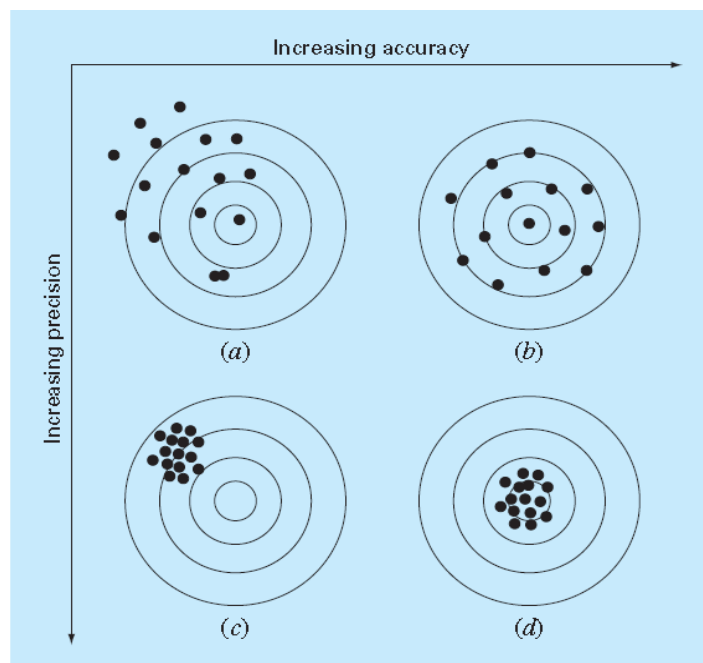
1. Significant digits,
2. Precision and accuracy,
3. Errors,
4. Number representation.

Significant digits

The **significant digits**, or figures, in a number are the digits of the number which are known to be correct. Engineering and scientific calculations generally begin with a set of data having a known number of significant digits. When these numbers are processed through a numerical algorithm, it is important to be able to estimate how many significant digits are present in the final computed result.

Precision and Accuracy

Precision refers to how closely a number represents the number it is representing. **Accuracy** refers to how closely a number agrees with the true value of the number it is representing. Precision is governed by the number of digits being carried in the numerical calculations. Accuracy is governed by the errors in a numerical calculation.



Errors

The accuracy of a numerical calculation is quantified by the **error** of the calculation. Several types of error can occur in numerical calculations.

1. Errors in the parameters of the problem (*assumed nonexistent*).
2. Algebraic errors in the calculations (*assumed nonexistent*).
3. Iteration errors.
4. Approximation errors.
5. Roundoff errors.
6. Truncation errors.

Iteration error is the error in an iterative method that approaches the exact solution of an exact problem asymptotically. Iteration errors must decrease toward zero as the iterative process progresses. The iteration error itself may be used to determine the successive approximations to the exact solution. Iteration error can be reduced the limit of the computing device. The errors in the solution of a system of linear algebraic equations by the successive over-relaxation (SOR) is an example of this type of errors.

Approximation error is the difference between the exact solution of an exact problem and the exact solution of an approximation of the exact problem. Approximation error can be reduced only by choosing a more accurate approximation of the exact problem. The error in the approximation of a function by a polynomial is an example of this type of errors. The error in the solution of a differential equation where the exact derivatives are replaced by algebraic difference approximations, which have truncation errors, is another example of this type of error.

Roundoff error is the error caused by the finite word length employed in the calculations. Roundoff error is more significant when small differences between large numbers are calculated. Most computers have either 32 bit or 64-bit word length, corresponding to approximately 7 or 13 significant decimal digits, respectively. Some computers have extended precision capability, which increases the number of bits to 128. Care must be exercised to ensure that enough significant digits are maintained in numerical calculations so that Roundoff is not significant.

For all error types, the relationship between the exact, or true, result and the approximation can be formulated as

$$\text{True value} = \text{approximation} + \text{error}$$

$$E_t = \text{true value} - \text{approximation}$$

where E_t is used to designate the exact value of the error. The subscript t is included to designate that this is the “true” error.

$$\text{True fractional relative error} = \frac{\text{true error}}{\text{true value}}$$

$$\varepsilon_t = \frac{\text{true error}}{\text{true value}} \times 100\%$$

where ε_t designates the true percent relative error.

Example:

Suppose that you have the task of measuring the lengths of a bridge and a rivet and come up with 9999 and 9 cm, respectively. If the true values are 10,000 and 10 cm, respectively, compute (a) the true error and (b) the true percent relative error for each case.

Solution:

(a) The error for measuring the bridge is

$$E_t = \text{true value} - \text{approximation}$$

$$E_t = 10,000 - 9,999 = 1 \text{ cm}$$

And for the rivet is:

$$E_t = 10 - 9 = 1 \text{ cm}$$

(b) The percent relative error for the bridge is

$$\varepsilon_t = \frac{\text{true error}}{\text{true value}} \times 100\%$$

$$\varepsilon_t = \frac{1}{10,000} \times 100\% = 0.01\%$$

And for the rivet is:

$$\varepsilon_t = \frac{1}{10} \times 100\% = 10\% .$$

However, in real-world applications, we will obviously not know the true answer a priori. For these situations, an alternative is to normalize the error using the best available estimate of the true value, that is, to the approximation itself, as in

$$\varepsilon_a = \frac{\text{approximate error}}{\text{approximate value}} \times 100\%$$

For **iterative approach**, the error is often estimated as the difference between previous and current approximations. Thus, percent relative error is determined according to

$$\varepsilon_a = \frac{\text{current approximation} - \text{previous approximation}}{\text{current approximation}} \times 100\%$$

It is also convenient to relate these errors to the number of significant figures in the approximation. It can be shown that if the following criterion is met, we can be assured that the result is correct to at least n significant figures.

$$\varepsilon_s = (0.5 \times 10^{2-n})\%$$

If this relationship holds, our result is assumed to be within the prespecified acceptable Level ε_s

Example:

For the following Maclaurin expansion of e^x , calculate the error iteratively for $x=0.5$ to three significant digits, knowing that $e^{x0.5} = 1.648721 \dots$

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$

Solution:

$$\varepsilon_s = (0.5 \times 10^{2-n})\% = (0.5 \times 10^{2-3})\% = 0.05\%$$

Taking the series one by one:

$$e^x = 1$$

$$e^x = 1 + x = 1 + 0.5 = 1.5$$

$$\varepsilon_t = \frac{\text{true error}}{\text{true value}} \times 100\% = \frac{1.648721 - 1.5}{1.648721} \times 100\% = 9.02\%$$

$$\varepsilon_a = \frac{\text{approximate error}}{\text{approximate value}} \times 100\% = \frac{1.5-1}{1.5} \times 100\% = 33.3\%$$

Because ε_a is not less than the required value of ε_s , we would continue the computation by adding another term, $\frac{x^2}{2!}$ and repeating the error calculations. The process is continued until ε_a , es. The entire computation can be summarized as

Terms	Result	ε_t (%)	ε_a (%)
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833333	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Truncation Errors and the Taylor Series

Truncation errors are those that result from using an approximation in place of an exact mathematical procedure. For example, the derivative of velocity of a falling parachutist may be approximated by a finite-divided-difference equation of the form:

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

A truncation error was introduced into the numerical solution because the difference equation only approximates the true value of the derivative

THE TAYLOR SERIES AND THE TAYLOR POLYNOMIAL

A power series in powers of x is a series of the form

$$\sum_{n=0}^{\infty} a_n x^n = a_0 + a_1 x + a_2 x^2 + \dots \quad (0.1)$$

A power series in powers of $(x - x_0)$ is given by

$$\sum_{n=0}^{\infty} a_n (x - x_0)^n = a_0 + a_1 (x - x_0) + a_2 (x - x_0)^2 + \dots \quad (0.2)$$

Within its radius of convergence, r , any continuous function, $f(x)$, can be represented exactly by a power series. Thus,

$$f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n \quad (0.3)$$

is continuous for $(x_0 - r) < x < (x_0 + r)$.

A. Taylor Series in One Independent Variable

If the coefficients, a_n , in Eq. (0.3) are given by the rule:

$$a_0 = f(x_0), a_1 = \frac{1}{1!} f'(x_0), a_2 = \frac{1}{2!} f''(x_0), \dots \quad (0.4)$$

then Eq. (0.3) becomes the Taylor series of $f(x)$ at $x = x_0$. Thus,

$$f(x) = f(x_0) + \frac{1}{1!} f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \dots \quad (0.5)$$

Equation (0.5) can be written in the simpler appearing form

$$f(x) = f_0 + f'_0 \Delta x + \frac{1}{2} f''_0 \Delta x^2 + \dots + \frac{1}{n!} f^{(n)}_0 \Delta x^n + \dots \quad (0.6)$$

where $f_0 = f(x_0)$, $f^{(n)} = d^n f / dx^n$, and $\Delta x = (x - x_0)$. Equation (0.6) can be written in the compact form

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} f^{(n)}_0 (x - x_0)^n$$

When $x_0 = 0$, the Taylor series is known as the **Maclaurin series**. In that case, Eqs. (0.5) and (0.7) become

$$f(x) = f(0) + f'(0)x + \frac{1}{2}f''(0)x^2 + \dots \quad (0.8)$$

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} f^{(n)}(0)x^n \quad (0.9)$$

It is, of course, impractical to evaluate an infinite Taylor series term by term. The Taylor series can be written as the finite Taylor series, also known as the *Taylor formula* or *Taylor polynomial* with remainder, as follows:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \dots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n + R^{n+1} \quad (0.10)$$

where the term R^{n+1} is the remainder term given by

$$R^{n+1} = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - x_0)^{n+1} \quad (0.11)$$

where ξ lies between x_0 and x . Equation (0.10) is quite useful in numerical analysis, where an approximation of $f(x)$ is obtained by truncating the remainder term.

B. Taylor Series in Two Independent Variables

Power series can also be written for functions of more than one independent variable. For a function of two independent variables, $f(x, y)$, the Taylor series of $f(x, y)$ at (x_0, y_0) is given by

$$f(x, y) = f_0 + \frac{\partial f}{\partial x} \Big|_0 (x - x_0) + \frac{\partial f}{\partial y} \Big|_0 (y - y_0) + \frac{1}{2!} \left(\frac{\partial^2 f}{\partial x^2} \Big|_0 (x - x_0)^2 + 2 \frac{\partial^2 f}{\partial x \partial y} \Big|_0 (x - x_0)(y - y_0) + \frac{\partial^2 f}{\partial y^2} \Big|_0 (y - y_0)^2 \right) + \dots \quad (0.12)$$

Equation (0.12) can be written in the general form

$$f(x, y) = \sum_{n=0}^{\infty} \frac{1}{n!} \left((x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y} \right)^n f(x, y) \Big|_0 \quad (0.13)$$

where the term $(\dots)^n$ is expanded by the binomial expansion and the resulting expansion operates on the function $f(x, y)$ and is evaluated at (x_0, y_0) .

The Taylor formula with remainder for a function of two independent variables is obtained by evaluating the derivatives in the $(n+1)$ st term at the point (ξ, η) , where (ξ, η) lies in the region between points (x_0, y_0) and (x, y) .

Example: Find Maclaurin for $\sin(x)$, $\cos(x)$, and e^x ?

Solution:

$$f(x) = f(0) + f'(0)x + \frac{1}{2!}f''(0)x^2 + \dots + \frac{1}{n!}f^{(n)}(0)x^n$$

$$f(x) = \sin x \quad f(0) = 0$$

$$f'(x) = \cos x \quad f'(0) = 1$$

$$f''(x) = -\sin x \quad f''(0) = 0$$

$$f'''(x) = -\cos x \quad f'''(0) = -1$$

$$f^{(4)}(x) = \sin x \quad f^{(4)}(0) = 0$$

$$f^{(5)}(x) = \cos x \quad f^{(5)}(0) = 1$$

$$f(x) = 0 + x + 0 - \frac{1}{3!}x^3 + 0 + \frac{1}{5!}x^5 + 0 - \frac{1}{7!}x^7 + \dots$$

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \dots + \frac{(-1)^{n-1}}{(2n-1)!}x^{2n-1}$$

$$\cos x = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots + \frac{(-1)^{n+1}}{(2n-2)!}x^{2n-2}$$

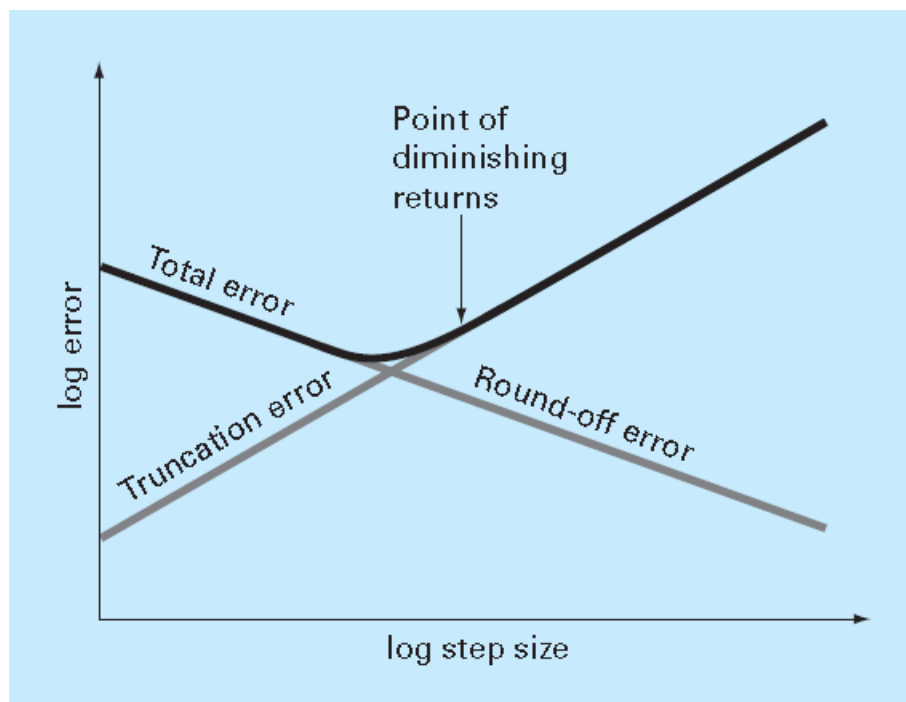
$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{n!}x^n$$

Example: Solve $\int_{1/2}^1 \frac{e^x}{x} dx$

Solution:

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{n!}x^n$$

$$\begin{aligned} \int_{1/2}^1 \frac{e^x}{x} dx &= \left(\frac{1}{x} + 1 + \frac{1}{2!}x + \frac{1}{3!}x^2 + \frac{1}{4!}x^3 + \dots \right) dx \\ &= \left[\ln x + x + \frac{1}{2 \times 2!}x^2 + \frac{1}{3 \times 3!}x^3 + \frac{1}{4 \times 4!}x^4 + \dots \right]_{1/2}^1 \end{aligned}$$



Consistency, Order, Smoothness and Convergence

There are several important concepts which must be considered when developing finite difference approximation of initial-value differential equations. they are (a) consistency, (b) order, (c) stability, and (d) convergence.

- *A **FDE** is consistent with an **ODE** if the difference between them (i.e., the truncation error) vanishes as $\Delta t \rightarrow 0$. In other words, the **FDE** approaches the **ODE**.*
- *The **order** of a **FDE** is the rate at which the global error decreases as the grid size approaches zero.*
- *A **FDE** is **stable** if it produces a bounded solution for a stable **ODE** and is **unstable** if it produces an unbounded solution for a stable **ODE**.*
- *A finite difference method is **convergent** if the numerical solution of the **FDE** (i.e., the numerical values) approaches the exact solution of the **ODE** as $\Delta t \rightarrow 0$*

MATLAB APPLICATIONS

```
function [v,ea,iter] = IterMeth(x,es,maxit)
% initialization e(x)
iter = 1;
sol = 1;
ea = 100;
% iterative calculation
while (1)
solold = sol;
sol = sol+x^iter/factorial(iter);
iter = iter+1;
if sol~=0
ea=abs((sol-solold)/sol)*100;
end
```

```
>> [val, ea, iter] = IterMeth(1,0.000001,100)
```

```
val = 2.7183
```

```
ea = 9.2162e-07
```

```
iter = 12
```

```

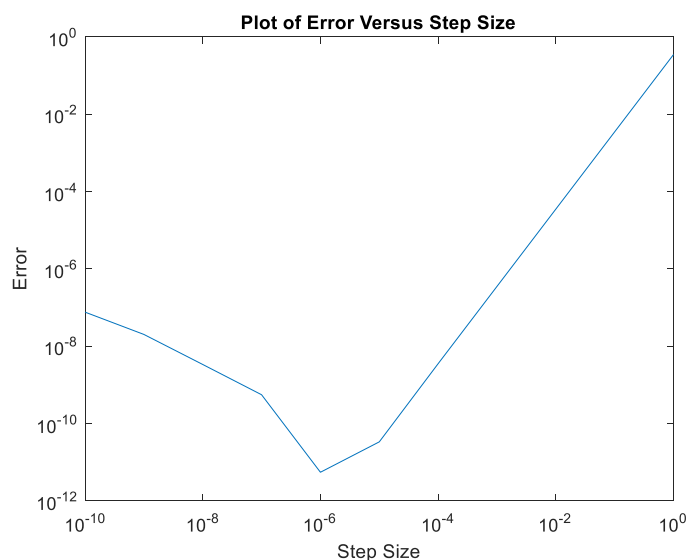
function diffex(func,dfunc,x,n)
format long
dftrue=dfunc(x);
h=1;
H(1)=h;
D(1)=(func(x+h)-func(x-h))/(2*h);
E(1)=abs(dftrue-D(1));
for i=2:n
h=h/10;
H(i)=h;
D(i)=(func(x+h)-func(x-h))/(2*h);
E(i)=abs(dftrue-D(i));
end

```

```

>> ff=@(x) -0.1*x^4-0.15*x^3-0.5*x^2-0.25*x+1.2;
>> df=@(x) -0.4*x^3-0.45*x^2-x-0.25;
>> diffex(ff,df,0.5,11)
step size finite difference true error
1.0000000000 -1.262500000000000 0.350000000000000
0.1000000000 -0.916000000000000 0.003500000000000
0.0100000000 -0.912535000000000 0.000035000000000
0.0010000000 -0.912500350000001 0.000000350000000
0.0001000000 -0.91250000349985 0.0000000034998
0.0000100000 -0.91250000003318 0.00000000000332
0.0000010000 -0.91250000000542 0.00000000000054
0.0000001000 -0.91249999945031 0.000000000005497
0.0000000100 -0.91250000333609 0.000000000033361
0.0000000010 -0.91250001998944 0.00000000199894
0.0000000001 -0.91250007550059 0.00000000755006

```



Numerical Analysis

DWE3214

PART-I: BASIC TOOLS

Unit-2: Roots of Nonlinear Equations



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-2: Roots of Nonlinear Equations

Introduction:

To solve $f(x) = ax^2 + bx + c = 0$ we usually use the quadratic formula (قانون الدستور)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \text{ to find the roots of the function that will make the function } f(x) = 0$$

Although the quadratic formula is handy for solving 2nd degree equations, there are many other functions for which the root cannot be determined so easily. For these cases, different numerical methods shall be provided as an efficient means to obtain the answer.

1- Closed (Bracketing) Methods

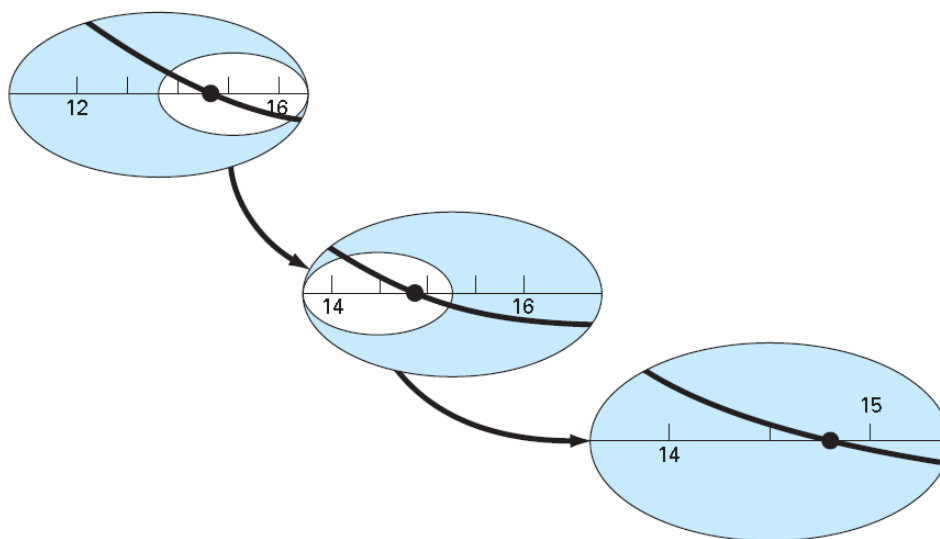
The Bisection Method:

Step 1: Choose lower x_L and upper x_U guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_L)f(x_U) < 0$.

Step 2: An estimate of the root x_R is determined by $x_R = \frac{x_L + x_U}{2}$.

Step 3: Make the following evaluations to determine in which subinterval the root lies:

- (a) If $f(x_L)f(x_R) < 0$, the root lies in the lower subinterval. Therefore, set $x_U = x_R$ and return to step 2.
- (b) If $f(x_L)f(x_R) > 0$, the root lies in the upper subinterval. Therefore, set $x_L = x_R$ and return to step 2.
- (c) If $f(x_L)f(x_R) = 0$, the root equals x_R ; terminate the computation.



Example:

Use the bisection method to determine the drag coefficient c needed for a parachutist of mass $m = 68.1$ kg to have a velocity of 40 m/s after freefalling for time $t = 10$ s?

Note: The acceleration due to gravity is 9.81 m/s².

Solution:

$$f(c) = \frac{9.81(68.1)}{c} \left(1 - e^{-\left(\frac{c}{68.1}\right)10} \right) - 40$$

or

$$f(c) = \frac{668.06}{c} (1 - e^{-0.146843c}) - 40$$

The first step in bisection is to guess two values of the unknown (in the present problem, c) that give values for $f(c)$ with different signs. We can see that the function changes sign between values of 12 and 16. Therefore, the initial estimate of the root x_R lies at the midpoint of the interval

$$x_R = \frac{12+16}{2} = 14 \quad (\varepsilon_t = 5.3\%)$$

$$f(12)f(14) = 6.114(1.611) = 9.850 > 0 \Rightarrow x_L = x_R = 14$$

Repeat the step:

$$x_R = \frac{14+16}{2} = 15 \quad (\varepsilon_t = 1.3\%)$$

$$f(14)f(15) = 1.611(-0.348) = -0.619 < 0 \Rightarrow x_U = x_R = 15$$

Repeat the step:

$$x_R = \frac{14+15}{2} = 14.5 \quad (\varepsilon_t = 2.0\%)$$

$$f(14)f(14.5) = 1.611(0.593) = 0.956 > 0 \Rightarrow x_L = x_R = 14.5$$

The method can be repeated until the result is accurate enough to satisfy your needs.

Note: the root of this equation is ($c = 14.8011$).

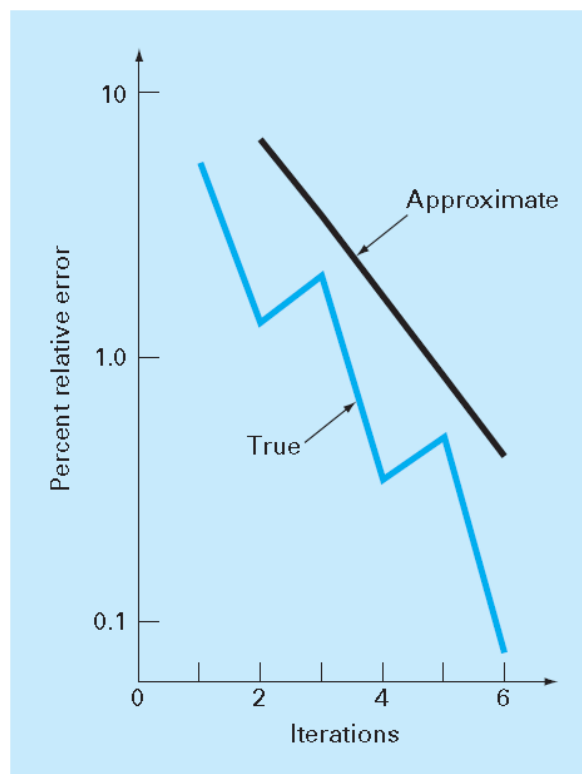
In this case we need a termination criterion:

$$\varepsilon_a = \left| \frac{x_R^{new} - x_R^{old}}{x_R^{new}} \right| 100\%$$

Each time we calculate the approximate error (because we do not know the true error) and stop the loop when the accuracy condition is satisfied.

For example: $\varepsilon_a = \left| \frac{15-14}{15} \right| 100\% = 6.667\%$ Recall that the true error was only 1.3%!

Iteration	x_l	x_u	x_r	ε_a (%)	ε_f (%)
1	12	16	14		5.413
2	14	16	15	6.667	1.344
3	14	15	14.5	3.448	2.035
4	14.5	15	14.75	1.695	0.345
5	14.75	15	14.875	0.840	0.499
6	14.75	14.875	14.8125	0.422	0.077



2- Open Methods

The Newton-Raphson Method:

The Newton-Raphson method can be derived on the basis of this geometrical interpretation (an alternative method based on the Taylor series).

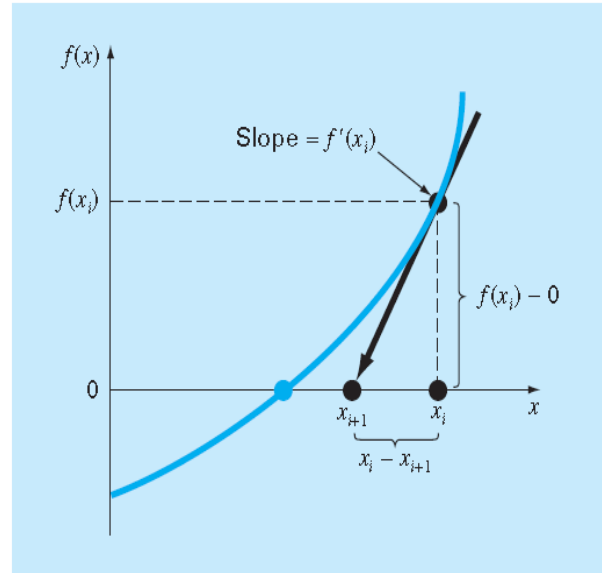
The first derivative at x is equivalent to the slope

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

the $f(x_{i+1}) = 0$ as can be seen from the figure

Which can be rearranged to yield

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



Example:

Use the Newton-Raphson method to estimate the root of $f(x) = e^{-x} - x$, employing an initial guess of $x_0 = 0$?

Solution:

$$\hat{f}(x) = -e^{-x} - 1$$

Newton-Raphson:

$$x_{i+1} = x_i - \frac{-e^{(-x_i)} - x_i}{-e^{(-x_i)} - 1}$$

i	x_i	$\epsilon_t(\%)$
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$< 10^{-8}$

Newton-Raphson Method for Systems of Non-linear Equations:

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ &\vdots \\ &\vdots \\ &\vdots \\ f_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

For example:

$$\begin{aligned} x^2 + xy &= 10 \\ \text{and} \quad y + 3xy^2 &= 57 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} u(x, y) &= x^2 + xy - 10 = 0 \\ v(x, y) &= y + 3xy^2 - 57 = 0 \end{aligned}$$

the solution would be the values of x and y that make the functions $u(x, y)$ and $v(x, y)$ equal to zero.

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)f'(x_i)$$

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

For system of non-linear equations:

$$u_{i+1} = u_i + (x_{i+1} - x_i) \frac{\partial u_i}{\partial x} + (y_{i+1} - y_i) \frac{\partial u_i}{\partial y}$$

and

$$v_{i+1} = v_i + (x_{i+1} - x_i) \frac{\partial v_i}{\partial x} + (y_{i+1} - y_i) \frac{\partial v_i}{\partial y}$$

Just as for the single-equation version, the root estimate corresponds to the values of x and y , where u_{i+1} and v_{i+1} equal zero leading to

$$\begin{aligned}\frac{\partial u_i}{\partial x}x_{i+1} + \frac{\partial u_i}{\partial y}y_{i+1} &= -u_i + x_i \frac{\partial u_i}{\partial x} + y_i \frac{\partial u_i}{\partial y} \\ \frac{\partial v_i}{\partial x}x_{i+1} + \frac{\partial v_i}{\partial y}y_{i+1} &= -v_i + x_i \frac{\partial v_i}{\partial x} + y_i \frac{\partial v_i}{\partial y}\end{aligned}$$

Consequently, algebraic manipulations (for example, Cramer's rule) can be employed to solve for

$$\begin{aligned}x_{i+1} &= x_i - \frac{u_i \frac{\partial v_i}{\partial y} - v_i \frac{\partial u_i}{\partial y}}{\frac{\partial u_i}{\partial x} \frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y} \frac{\partial v_i}{\partial x}} \\ y_{i+1} &= y_i - \frac{v_i \frac{\partial u_i}{\partial x} - u_i \frac{\partial v_i}{\partial x}}{\frac{\partial u_i}{\partial x} \frac{\partial v_i}{\partial y} - \frac{\partial u_i}{\partial y} \frac{\partial v_i}{\partial x}}\end{aligned}$$

The denominator of each of these equations is formally referred to as the determinant of the **Jacobian** of the system.

Example: Solve the following system of non-linear equations, Initiate the computation with guesses of $x = 1.5$ and $y = 3.5$?

$$\begin{aligned}u(x, y) &= x^2 + xy - 10 = 0 \\ v(x, y) &= y + 3xy^2 - 57 = 0\end{aligned}$$

Solution:

First compute the partial derivatives and evaluate them at the initial guesses of x and y :

$$\begin{aligned}\frac{\partial u_0}{\partial x} &= 2x + y = 2(1.5) + 3.5 = 6.5 & \frac{\partial u_0}{\partial y} &= x = 1.5 \\ \frac{\partial v_0}{\partial x} &= 3y^2 = 3(3.5)^2 = 36.75 & \frac{\partial v_0}{\partial y} &= 1 + 6xy = 1 + 6(1.5)(3.5) = 32.5\end{aligned}$$

Thus, the determinant of the Jacobian for the first iteration is

$$6.5(32.5) - 1.5(36.75) = 156.125$$

The values of the functions can be evaluated at the initial guesses as

$$u_0 = (1.5)^2 + 1.5(3.5) - 10 = -2.5$$

$$v_0 = 3.5 + 3(1.5)(3.5)^2 - 57 = 1.625$$

These values can be substituted into the equation to give

$$x = 1.5 - \frac{-2.5(32.5) - 1.625(1.5)}{156.125} = 2.03603$$

$$y = 3.5 - \frac{1.625(6.5) - (-2.5)(36.75)}{156.125} = 2.84388$$

Thus, the results are converging to the true values of $x = 2$ and $y = 3$. The computation can be repeated until an acceptable accuracy is obtained.

Numerical Analysis

DWE3214

PART-I: BASIC TOOLS

Unit-3: Simultaneous Linear algebraic Equations



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-3: Simultaneous Linear algebraic Equations

DETERMINANTS

For every matrix of order n , i.e., for every $n \times n$ matrix A , the corresponding value of the determinant function is denoted by either $|A|$ or $\det A$.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \Leftrightarrow \text{matrix } A$$

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \Leftrightarrow \det A$$

The determinant $|M|$ formed by the m^2 elements common to any m rows and any m columns of an n th-order determinant $|A|$ is said to be an m^{th} -order minor of $|A|$. The determinant of order $n - m$ formed by the array of elements which remains when the m rows and m columns containing an m^{th} -order minor $|M|$ are deleted from $|A|$ is called the complementary minor of $|M|$.

If the numbers of the rows and columns of $|A|$ which contain an m^{th} -order minor $|M|$ are, respectively,

$$i_1, i_2, \dots, i_m \quad \text{and} \quad j_1, j_2, \dots, j_m$$

then $(-1)^{i_1+i_2+\dots+i_m+j_1+j_2+\dots+j_m}$ times the complementary minor of $|M|$ is called the algebraic complement and/or (cofactors) of $|M|$.

We shall denote the complementary minor of the element a_{ij} by the symbol M_{ij} and its algebraic complement (cofactor) by the symbol A_{ij} ; thus

$$A_{ij} = (-1)^{i+j} M_{ij} \text{ which is for the first order}$$

And for the second order complementary minor and its algebraic complement (cofactor) we use

$$A_{ij,kl} = (-1)^{i+j+k+l} M_{ij,kl} \text{ where } i,j \text{ are the rows and } k,l \text{ are the columns.}$$

Example: In the fifth-order determinant,

$$|A| = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{vmatrix}$$

the complementary minor of the element a_{43} is the fourth-order determinant formed by the elements which remains when the fourth row and the third column are deleted from $|A|$, namely,

$$M_{4,3} = \begin{vmatrix} a_{11} & a_{12} & \vdots & a_{14} & a_{15} \\ a_{21} & a_{22} & \vdots & a_{24} & a_{25} \\ a_{31} & a_{32} & \vdots & a_{34} & a_{35} \\ \dots & \dots & \dots & \dots & \dots \\ a_{51} & a_{52} & \vdots & a_{54} & a_{55} \end{vmatrix} \Rightarrow M_{4,3} = \begin{vmatrix} a_{11} & a_{12} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{34} & a_{35} \\ a_{51} & a_{52} & a_{54} & a_{55} \end{vmatrix}$$

the cofactor $A_{4,3}$ of the element $a_{4,3}$ is equal to this complementary minor times $(-1)^{4+3}$; that is,

$$A_{4,3} = (-1)^{4+3} M_{4,3} = -M_{4,3}$$

Similarly, the complementary minor of the second-order minor $a_{35,14} = \begin{vmatrix} a_{31} & a_{34} \\ a_{51} & a_{54} \end{vmatrix}$ contained in the third and fifth rows and the first and fourth of $|A|$ is the third order determinant formed by the elements which remain when these rows and columns are deleted from $|A|$:

$$M_{35,14} = \begin{vmatrix} \vdots & a_{12} & a_{13} & \vdots & a_{15} \\ \vdots & a_{22} & a_{23} & \vdots & a_{25} \\ \dots & \dots & \dots & \dots & \dots \\ \vdots & a_{42} & a_{43} & \vdots & a_{45} \\ \dots & \dots & \dots & \dots & \dots \end{vmatrix} \Rightarrow M_{35,14} = \begin{vmatrix} a_{12} & a_{13} & a_{15} \\ a_{22} & a_{23} & a_{25} \\ a_{42} & a_{43} & a_{45} \end{vmatrix}$$

The algebraic complement (cofactor) $A_{35,14}$ of the given second-order minor is equal to the complementary minor $M_{35,14}$ times $(-1)^{3+5+1+4}$; that is,

$$A_{35,14} = (-1)^{3+5+1+4} M_{35,14} \Rightarrow A_{35,14} = -M_{35,14}$$

DEFINITION 1: The determinant of a matrix with a single element is that element. For every matrix A of order $n \geq 2$,

$$\det A = a_{11} A_{11} + a_{12} A_{12} + \dots + a_{1n} A_{1n} = \sum_{k=1}^n a_{1k} A_{1k}$$

Simplification for 2nd and 3rd order determinants:

For 2nd order matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$\det A = a_{11} a_{22} - a_{12} a_{21}$$

$$\begin{vmatrix} (+) & & & & \\ & \ddots & & & \\ & & a_{11} & & a_{12} \\ & & & \ddots & \\ & & a_{21} & & a_{22} \\ & \ddots & & & \\ (-) & & & & \end{vmatrix} = a_{11} a_{22} - a_{12} a_{21}$$

For 3rd order matrix:

A general third-order determinant can be expanded using the former equations:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

$$= a_{11} [a_{22} a_{33} - a_{23} a_{32}] - a_{12} [a_{21} a_{33} - a_{23} a_{31}] + a_{13} [a_{21} a_{32} - a_{22} a_{31}]$$

$$= a_{11} a_{22} a_{33} + a_{12} a_{23} a_{31} + a_{13} a_{21} a_{32} - a_{13} a_{22} a_{31} - a_{11} a_{23} a_{32} - a_{12} a_{21} a_{33}$$

The expansion can also be obtained by diagonal multiplication, by repeating on the right the first two columns of the determinate and then adding the signed products of the elements on the various diagonals in the resulting array:

$$\begin{vmatrix}
 \ddots & (+) & \ddots & (+) & \ddots & (+) \\
 & \ddots & & \ddots & & \ddots \\
 & & a_{11} & & a_{12} & & a_{13} \\
 & & & \ddots & & \ddots & \\
 & & a_{21} & & a_{22} & & a_{23} \\
 & & & \ddots & & \ddots & \\
 & & a_{31} & & a_{32} & & a_{33} \\
 & \ddots & & \ddots & & \ddots & \\
 \ddots & (-) & \ddots & (-) & \ddots & (-)
 \end{vmatrix}
 \begin{vmatrix}
 & & a_{11} & & a_{12} \\
 & & & \ddots & \\
 a_{21} & & a_{22} \\
 & & & \ddots & \\
 a_{31} & & a_{32}
 \end{vmatrix}$$

PROPERTY 1: For every matrix A of order n , and for each i such that $1 \leq i \leq n$,

$$\det A = \sum_{k=1}^n a_{ik} A_{ik}$$

and for each $1 \leq j \leq n$

$$A = \sum_{k=1}^n a_{jk} A_{jk}$$

Which means that we can use any row or column to evaluate the determinate and of course any row or column with a lot of zero element is preferred.

Example: Evaluate the forth-order determinant

$$|A| = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 0 & -1 & 2 & 3 \\ 1 & 6 & 4 & -2 \end{vmatrix}$$

Solution: Using the third row

$$|A| = (0) \begin{vmatrix} 2 & 3 & 4 \\ 3 & 2 & 1 \\ 6 & 4 & -2 \end{vmatrix} - (-1) \begin{vmatrix} 1 & 3 & 4 \\ 4 & 2 & 1 \\ 1 & 4 & -2 \end{vmatrix} + (2) \begin{vmatrix} 1 & 2 & 4 \\ 4 & 3 & 1 \\ 1 & 6 & -2 \end{vmatrix} - (3) \begin{vmatrix} 1 & 2 & 3 \\ 4 & 3 & 2 \\ 1 & 6 & 4 \end{vmatrix}$$

$$|A| = 0 + 75 + 180 - 105 = 150$$

we can obtain the same result easily by using the first column

PROPERTY 2: For every square matrix A , $\det A^T = \det A$.

Example: for the matrix $A = \begin{bmatrix} 1 & 0 & 0 \\ 5 & -2 & 0 \\ 9 & 14 & 3 \end{bmatrix}$ show that $\det A^T = \det A$.

Solution: Expanding successive determinants by elements of their first rows, we find:

$$\det A = \begin{vmatrix} 1 & 0 & 0 \\ 5 & -2 & 0 \\ 9 & 14 & 3 \end{vmatrix} = (1) \begin{vmatrix} -2 & 0 \\ 14 & 3 \end{vmatrix} = (1)(-2)(3) = -6$$

Expanding successive determinants by elements of their first columns, we find:

$$\det A^T = \begin{vmatrix} 1 & 5 & 9 \\ 0 & -2 & 14 \\ 0 & 0 & 3 \end{vmatrix} = (1) \begin{vmatrix} -2 & 14 \\ 0 & 3 \end{vmatrix} = (1)(-2)(3) = -6 = \det A$$

Both A and A^T , in the previous example, are triangular matrices and the determinant of either matrix is equal to the product of its diagonal elements.

PROPERTY 3: If A is a triangular matrix of order n , then

$$\det A = a_{11} a_{22} \dots a_{nn} \Rightarrow \det A = \prod_{i=1}^n a_{i,i}$$

that is to say the determinant of a triangular matrix is equal to the product of its diagonal elements.

See the previous example.

PROPERTY 4: If a square matrix A has either a zero row or a zero column, then:

$$\det A = 0$$

Example: Find $|A|$ of $A = \begin{bmatrix} 5 & 0 & 3 \\ -6 & 0 & 6 \\ 4 & 0 & 2 \end{bmatrix}$

Solution: Applying the expansion by elements of the second column, we find that

$$\begin{vmatrix} 5 & 0 & 3 \\ -6 & 0 & 6 \\ 4 & 0 & 2 \end{vmatrix} = -(0) \begin{vmatrix} -6 & 6 \\ 4 & 2 \end{vmatrix} + (0) \begin{vmatrix} 5 & 3 \\ 4 & 2 \end{vmatrix} - (0) \begin{vmatrix} 5 & 3 \\ -6 & 6 \end{vmatrix} = 0$$

PROPERTY 5: If each element in one row or column of a determinant is multiplied by a number c , the value of the determinant is multiplied by c too.

Example: Evaluate if the 2nd row is factored by 2 and the 2nd column is factored by 3?

$$\begin{vmatrix} 3 & 9 & 5 \\ 4 & 6 & 0 \\ -1 & -3 & 2 \end{vmatrix}$$

Solution:

First, we evaluate the determinant and then multiply by (2×3) :

$$2 \times 3 \begin{vmatrix} 3 & 9 & 5 \\ 4 & 6 & 0 \\ -1 & -3 & 2 \end{vmatrix} = 6 \begin{vmatrix} 3 & 9 \\ 4 & 6 \\ -1 & -3 \end{vmatrix} = 6(36 + 0 - 60 + 30 - 0 - 72) = 6(-66) = -396$$

Second, we multiply the determinants by the factors and solve:

$$2 \times 3 \begin{vmatrix} 3 & 9 & 5 \\ 4 & 6 & 0 \\ -1 & -3 & 2 \end{vmatrix} = \begin{vmatrix} 3 & 27 & 5 \\ 8 & 36 & 0 \\ -1 & -9 & 2 \end{vmatrix}$$

$$\begin{vmatrix} 3 & 27 & 5 \\ 8 & 36 & 0 \\ -1 & -9 & 2 \end{vmatrix} = \begin{vmatrix} 3 & 27 \\ 8 & 36 \\ -1 & -9 \end{vmatrix} = (216 + 0 - 360 + 180 - 0 - 432) = -396 \quad \text{we are done}$$

PROPERTY 6: If $A = [v_1 \dots f_j + g_j \dots v_n]$ is a square matrix, then
 $\det A = \det [v_1 \dots f_j \dots v_n] + \det [v_1 \dots g_j \dots v_n]$

What this property says is that, if each element in one column (row) of a determinant is expressed as a binomial, the determinant can be written as the sum of two determinants. Specifically,

Example: Determine the number k such that

$$\begin{vmatrix} 1 & 2 & -3 \\ -k & 1+3k & 3-k \\ 0 & -6 & 5 \end{vmatrix} = 36$$

Solution:

With $-k$ regarded as the binomial $0-k$, the elements of the second row of the given determinant become binomial; hence

$$\begin{vmatrix} 1 & 2 & -3 \\ -k & 1+3k & 3-k \\ 0 & -6 & 5 \end{vmatrix} = \begin{vmatrix} 1 & 2 & -3 \\ 0 & 1 & 3 \\ 0 & -6 & 5 \end{vmatrix} + \begin{vmatrix} 1 & 2 & -3 \\ -k & 3k & -k \\ 0 & -6 & 5 \end{vmatrix} = 23 + k \begin{vmatrix} 1 & 2 & -3 \\ -1 & 3 & -1 \\ 0 & -6 & 5 \end{vmatrix} = 23 + k$$

Which equals 36 if and only if $k=13$.

PROPERTY 7: If B is a matrix obtained by interchanging any two rows (columns) of a square matrix A , $\det B = -\det A$.

PROPERTY 8: If one row (column) vector of a square matrix A is equal to a number c times some other row (column) vector, then $|A| = 0$

Example: Evaluate the determinant

$$|A| = \begin{vmatrix} 2 & -3 & 2 \\ 3 & -2 & 5 \\ -6 & 9 & 6 \end{vmatrix}$$

Solution:

$$\begin{vmatrix} 2 & -3 & 2 \\ 3 & -2 & 5 \\ -6 & 9 & 6 \end{vmatrix} = 0$$

it is apparent that each element of the third row is equal to -3 times the corresponding element of the first row; hence, $|A| = 0$

PROPERTY 9: If a matrix **B** is obtained from a square matrix **A** by adding to one row (column) vector of **A** a number *c* times a different row (column) vector, then $\det B = \det A$.

Hint: this property is very useful especially with large determinants.

Example: Find the value of the determinant

$$\begin{vmatrix} 3 & 1 & -1 & 2 & 1 \\ 0 & 3 & 1 & 4 & 2 \\ 1 & 4 & 2 & 3 & 1 \\ 5 & -1 & -3 & 2 & 5 \\ -1 & 1 & 2 & 3 & 2 \end{vmatrix}$$

Solution:

To introduce as many zeros as possible in some rows (columns):

- 1) Add the 3rd column to the 2nd and the 5th.
- 2) Add twice the 3rd column to the 4th.
- 3) Add 3 times the 3rd column to the 1st.

This gives the new but equal determinant:

$$\begin{array}{ccccc} 0 & 0 & -1 & 0 & 0 \\ 3 & 4 & 1 & 6 & 3 \\ 7 & 6 & 2 & 7 & 3 \\ -4 & -4 & -3 & -4 & 2 \\ 5 & 3 & 2 & 7 & 4 \end{array}$$

Expanding this in terms of the first row, according to property 1, we have:

$$\det A = \sum_{k=1}^n a_{ik} A_{ik}$$

$$A_{ij,kl} = (-1)^{i+j+k+l} M_{ij,kl}$$

$$\det A = 0 + 0 + (-1)(-1)^{1+3} \begin{vmatrix} 3 & 4 & 6 & 3 \\ 7 & 6 & 7 & 3 \\ -4 & -4 & -4 & 2 \\ 5 & 3 & 7 & 4 \end{vmatrix} + 0 + 0$$

Now to simplify the 4th order determinant too:

Add twice the last column to each of the first three; we obtain the equal determinant,

$$- \begin{vmatrix} 9 & 10 & 12 & 3 \\ 13 & 12 & 13 & 3 \\ 0 & 0 & 0 & 2 \\ 13 & 11 & 15 & 4 \end{vmatrix} \text{ and expanding in terms of the } 3^{\text{rd}} \text{ row,}$$

$$- (2)(-1)^{3+4} \begin{vmatrix} 9 & 10 & 12 \\ 13 & 12 & 13 \\ 13 & 11 & 15 \end{vmatrix}$$

We can now simplify this by further row or column manipulations, or, since it is of the 3rd order, we can expand it by the diagonal method, the result is -166.

PROPERTY 10: If A and B are matrices of the same order, then

$$(\det A)(\det B) = \det(AB)$$

Example: For the matrices:

$$A = \begin{bmatrix} 1 & 0 & -3 \\ 2 & -5 & 4 \\ -2 & 3 & -1 \end{bmatrix} \quad B = \begin{bmatrix} -3 & -2 & 12 \\ 4 & 1 & -6 \\ 2 & 3 & -10 \end{bmatrix}$$

We verify directly that $(\det A)(\det B) = \det(AB)$

Add 3 times the first column of $\det A$ to its 3rd column, we have:

$$\det A = \begin{vmatrix} 1 & 0 & 0 \\ 2 & -5 & 10 \\ -2 & 3 & -7 \end{vmatrix} = (-1)^2 \begin{vmatrix} -5 & 10 \\ 3 & -1 \end{vmatrix} = 35 - 30 = 5$$

Add 2 times the 2nd row to the 1st row, and -3 times 2nd row to the 3rd row, of $\det B$, we find,

$$\det B = \begin{vmatrix} 5 & 0 & 0 \\ 4 & 1 & -6 \\ -10 & 0 & 8 \end{vmatrix} = 5 \begin{vmatrix} 1 & -6 \\ 0 & 8 \end{vmatrix} = 5 \times 8 = 40$$

We next compute the matrix product AB from which we obtain:

$$\begin{aligned} \det(AB) &= \begin{vmatrix} -9 & -11 & 42 \\ -18 & 3 & 14 \\ 16 & 4 & -32 \end{vmatrix} = 4 \begin{vmatrix} -9 & -11 & 42 \\ -18 & 3 & 14 \\ 4 & 1 & -8 \end{vmatrix} = 4 \begin{vmatrix} 35 & -11 & -46 \\ -30 & 3 & 38 \\ 0 & 1 & 0 \end{vmatrix} \\ &= (4)(-1)^{3+2} \begin{vmatrix} 35 & -46 \\ -30 & 38 \end{vmatrix} = -(4 \times 5 \times 2) \begin{vmatrix} 7 & -23 \\ -6 & 19 \end{vmatrix} = -40(133 - 138) = 200 \\ &= \det A \times \det B \end{aligned}$$

MATRICES

The fundamental quantity of linear algebra is the matrix. A matrix is an ordered rectangular array of numbers or mathematical expressions. We shall use upper case letters to denote them. The $m \times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdot & \cdot & \cdot & a_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & a_{ij} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & a_{m3} & \cdot & \cdot & \cdot & a_{mn} \end{bmatrix}$$

has m rows and n columns. If $m=n$, the matrix is a square matrix; otherwise, A is rectangular. For a square matrix, the diagonal from the top left corner to the bottom right corner is the principal diagonal.

From the limitless number of possible matrices, certain one appears with sufficient regularity that they are given special names.

Zero Matrix: sometimes called a null matrix, has all of its elements equal to zero.

Unit Matrix: the unit of identity matrix is a $n \times n$ matrix having 1's along the principal diagonal and zero everywhere else.

Symmetric Matrix: a symmetric matrix is one where $a_{ij} = a_{ji}$ for all i and j .

Example: Examples of zero, Identity, and symmetric matrices are

$$O = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 3 & 2 & 4 \\ 2 & 1 & 0 \\ 4 & 0 & 5 \end{pmatrix}$$

respectively.

Diagonal Matrix: a diagonal matrix is a $n \times n$ matrix having values along the principal diagonal and zero everywhere else.

Upper Triangular Matrix: an upper triangular matrix is a $n \times n$ matrix having **values** along the principal diagonal and the upper triangle. And zeros in the lower triangle.

Lower Triangular Matrix: lower triangular matrix is a $n \times n$ matrix having **values** along the principal diagonal and the lower triangle. And zeros in the upper triangle.

Example: Examples of Diagonal, Upper Triangular, and Lower Triangular Matrices are

$$D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 0 & 6 & 4 \\ 0 & 5 & 9 & 7 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 3 \end{pmatrix}, \quad L = \begin{pmatrix} 7 & 0 & 0 \\ 2 & -3 & 0 \\ 1 & 10 & 1 \end{pmatrix}$$

respectively.

Definition 1: Two matrices A and B are equal if and only if $a_{ij} = b_{ij}$ for all possible i and j and they have the same dimensions.

Definition 2: For two matrices A and B with the same dimensions (conformable for addition), the matrix $C = A + B$ contains the elements $c_{ij} = a_{ij} + b_{ij}$. Similarly, $C = A - B$ contains the elements $c_{ij} = a_{ij} - b_{ij}$. Because the order of addition does not matter, addition is cumulative: $A + B = B + A$.

Definition 3: considering a scalar constant k . The product $k \times A$ is formed by multiplying every element of A by k . thus the matrix kA has elements equal to $k \times a_{ij}$.

Matrix Multiplication:

We begin by requiring that the dimension of A be $m \times n$ while for B they are $n \times p$. That is, the number of columns in A must equal the number of rows in B . The matrices A and B are then said to be conformable for multiplication. If this is true, then $C = A \times B$ will be a matrix $m \times p$, where its elements equal

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}.$$

The right side of the equation is referred to as an inner product of the i^{th} row of A and the j^{th} column of B . The product $A \times A$ is usually written A^2 ; the product $A \times A \times A$ is usually written A^3 ; and so forth.

Example: If

$$A = \begin{bmatrix} -1 & 4 \\ 2 & -3 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

then

$$\begin{aligned} AB &= \begin{pmatrix} [(-1)(1) + (4)(3)] & [(-1)(2) + (4)(4)] \\ [(2)(1) + (-3)(3)] & [(2)(2) + (-3)(4)] \end{pmatrix} \\ &= \begin{pmatrix} 11 & 14 \\ -7 & -8 \end{pmatrix} \end{aligned}$$

Matrix multiplication is associative and distributive with respect to addition:

$$(kA) B = k (AB) = A (kB),$$

$$A (BC) = (AB) C,$$

$$(A+B) C = AC + BC$$

$$C (A+B) = CA + CB$$

On the other hand, matrix multiplication is not cumulative. In general, $AB \neq BA$.

Example: Does $AB = BA$ if

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad ?$$

Because

$$AB = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

and

$$BA = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

$$AB \neq BA.$$

Matrix Transposition:

Another matrix operation is transposition. The **transpose** of a matrix A with dimensions $m \times n$ is another matrix, written A^T , where we have interchanged the rows and columns from A . Clearly, $(A^T)^T = A$ as well as $(A+B)^T = A^T + B^T$ and $(kA)^T = kA^T$. If A and B are conformable for multiplication, then $(AB)^T = B^T A^T$. Note the reversal of order between the two sides

Example: Given A , find A^T ?

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 10 & 5 & 1 \\ 6 & 7 & 8 \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} 2 & 10 & 6 \\ 3 & 5 & 7 \\ 4 & 1 & 8 \end{bmatrix}$$

Matrix Inverse:

A matrix A is said to be non-singular or invertible if there exists a matrix B such that $AB = BA = I$. This matrix B is the multiplicative inverse of A or simply the inverse of A , written A^{-1} . An $n \times n$ matrix is singular if it does not have a multiplicative inverse.

From preliminary Algebra we know that the inverse of any quantity Q is $\frac{1}{Q}$ which is usually denoted as Q^{-1} .

$$Q^{-1} = \frac{1}{Q}$$

$$Q \times Q^{-1} = 1$$

The same equation is applicable for square matrix A ,

$$[A][A^{-1}] = I$$

$$[A^{-1}] = \frac{\text{adj } A}{|A|} \quad \text{where,}$$

$|A|$ is the determinant of the matrix A

$\text{adj } A$ is the adjoint of the matrix A

$$\text{adj } A = [A_{ij}]^T \quad \text{where,}$$

A_{ij} is the algebraic complementary (cofactor) of the element a_{ij}

Example: Find A^{-1} if

$$A = \begin{bmatrix} 1 & 2 & 4 \\ -1 & 0 & 3 \\ 3 & 1 & -2 \end{bmatrix}$$

$$|A| = \begin{vmatrix} 1 & 2 & 4 \\ -1 & 0 & 3 \\ 3 & 1 & -2 \end{vmatrix} \quad -2R_3 + R_1 \Rightarrow$$

$$|A| = \begin{vmatrix} -5 & 0 & 8 \\ -1 & 0 & 3 \\ 3 & 1 & -2 \end{vmatrix} = 1 \times (-1)^{3+2} \begin{vmatrix} -5 & 8 \\ -1 & 3 \end{vmatrix} = (-1)(-15 + 8) = 7$$

$$\text{adj } A = \begin{bmatrix} (-1)^{1+1} \begin{vmatrix} 0 & 3 \\ 1 & -2 \end{vmatrix} & (-1)^{1+2} \begin{vmatrix} -1 & 3 \\ 3 & -2 \end{vmatrix} & (-1)^{1+3} \begin{vmatrix} -1 & 0 \\ 3 & 1 \end{vmatrix} \\ (-1)^{2+1} \begin{vmatrix} 2 & 4 \\ 1 & -2 \end{vmatrix} & (-1)^{2+2} \begin{vmatrix} 1 & 4 \\ 3 & -2 \end{vmatrix} & (-1)^{2+3} \begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix} \\ (-1)^{3+1} \begin{vmatrix} 2 & 4 \\ 0 & 3 \end{vmatrix} & (-1)^{3+2} \begin{vmatrix} 1 & 4 \\ -1 & 3 \end{vmatrix} & (-1)^{3+3} \begin{vmatrix} 1 & 2 \\ -1 & 0 \end{vmatrix} \end{bmatrix}^T$$

$$\text{adj } A = \begin{bmatrix} -3 & 8 & 6 \\ 7 & -14 & -7 \\ -1 & 5 & 2 \end{bmatrix}$$

$$A^{-1} = \frac{\text{adj } A}{|A|} = \frac{1}{7} \begin{bmatrix} -3 & 8 & 6 \\ 7 & -14 & -7 \\ -1 & 5 & 2 \end{bmatrix}$$

Check

$$[A \times A^{-1}] = \frac{1}{7} \begin{bmatrix} -3 & 8 & 6 \\ 7 & -14 & -7 \\ -1 & 5 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 4 \\ -1 & 0 & 3 \\ 3 & 1 & -2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

SYSTEMS OF LINEAR ALGEBRAIC EQUATIONS

Systems of linear algebraic equations can be expressed very compactly in matrix notation, such as,

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

Or can be written

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad (i = 1, \dots, n)$$

There are three so-called **row operations** that are useful when solving systems of linear algebraic equations. They are:

1. Any row (equation) may be multiplied by a constant (a process known as **scaling**).
2. The order of the rows (equations) may be interchanged (a process known as **pivoting**).
3. Any row (equation) can be replaced by a weighted linear combination of that row (equation) with any other row (equation) (a process known as **elimination**).

In the context of the solution of a system of linear algebraic equations, these three row operations clearly do not change the solution.

1. Direct Elimination Methods

Cramer's Rule

Although it is not an elimination method, Cramer's rule is a direct method for solving systems of linear algebraic equations. Consider the system of linear algebraic equations, $A\mathbf{x} = \mathbf{b}$, which represents n equations. Cramer's rule states that the solution for x_j ($j=1, \dots, n$) is given by:

$$x_j = \frac{\det(A^j)}{\det(A)} \quad j = 1, \dots, n$$

where $[A^j]$ is $n \times n$ matrix obtained by replacing column j in matrix A by the column vector \mathbf{b} . For example, consider the system of two linear algebraic equations:

$$a_{11} x_1 + a_{12} x_2 = b_1$$

$$a_{21} x_1 + a_{22} x_2 = b_2$$

Applying Cramer's rule yields

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} \quad \text{and} \quad x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$$

The determinants in the last equation can be evaluated by the diagonal method.

For systems containing more than three equations, the diagonal method does not work. In such cases, the method of cofactors could be used. The number of multiplications and divisions N required by the method of cofactors is $N = (n-1)(n+1)!$. For a relatively small system of 10 equations (i.e., $n=10$), $N=360,000,000$, which is an enormous number of calculations. For $n=100$, $N=10^{157}$, which is obviously ridiculous. The preferred method for evaluating determinant is elimination method. The number of multiplications and divisions required by the elimination method is approximately $N = n^3 + n^2 - n$. Thus, for $n=10$, $N=1090$, and for $n=100$, $N=1,009,900$. Obviously, the elimination method is preferred.

Example: Cramer's rule.

Let's illustrate Cramer's rule by solving the following system,

$$\begin{aligned}80x_1 - 20x_2 - 20x_3 &= 20 \\ -20x_1 + 40x_2 - 20x_3 &= 20 \\ -20x_1 - 20x_2 + 130x_3 &= 20\end{aligned}$$

Solution:

$$\det(A) = \begin{vmatrix} 80 & -20 & -20 \\ -20 & 40 & -20 \\ -20 & -20 & 130 \end{vmatrix} = 300,000$$

Next calculate $\det(A^1)$, $\det(A^2)$, and $\det(A^3)$.

$$\det(A^1) = \begin{vmatrix} 20 & -20 & -20 \\ 20 & 40 & -20 \\ 20 & -20 & 130 \end{vmatrix} = 180,000$$

$$\det(A^2) = \begin{vmatrix} 80 & 20 & -20 \\ -20 & 20 & -20 \\ -20 & 20 & 130 \end{vmatrix} = 300,000$$

$$\det(A^3) = \begin{vmatrix} 80 & -20 & 20 \\ -20 & 40 & 20 \\ -20 & -20 & 20 \end{vmatrix} = 120,000$$

$$x_1 = \frac{\det(A^1)}{\det(A)} = \frac{180,000}{300,000} = 0.60 \quad x_2 = \frac{\det(A^2)}{\det(A)} = \frac{300,000}{300,000} = 1.00 \quad x_3 = \frac{\det(A^3)}{\det(A)} = \frac{120,000}{300,000} = 0.40$$

Gauss Elimination

Elimination methods solve a system of linear algebraic equations by solving one equation; say the first equation, for one of the unknowns, say x_1 , in terms of the remaining unknowns, x_2 to x_n , then substituting the expression for x_1 into the remaining $n-1$ equations to determine $n-1$ equations involving x_2 to x_n . This elimination procedure is performed $n-1$ times until the last step yields an equation involving only x_n . This process is called **elimination**.

The value of x_n can be calculated from the final equation in the elimination procedure. Then x_{n-1} can be calculated from modified equation $n-1$, which contains only x_n and x_{n-1} . then x_{n-2} can be calculated from modified equation $n-2$, which contains only x_n , x_{n-1} , and x_{n-2} . This procedure is performed $n-1$ times to calculate x_{n-1} to x_1 . This process is called **back substitution**.



Johann Carl Friedrich Gauss

1777 - 1855

The elimination procedure described in the previous section, including scaled pivoting, is commonly called **Gauss elimination**. It is the most important and most useful direct elimination method for solving systems of linear algebraic equations. The **Gauss-Jordan** method, the **matrix inverse** method, the **LU factorization** method, and the **Thomas algorithm** are all modifications or extensions of Gauss elimination method. **Pivoting** is an essential element of **Gauss elimination**.

Elimination

Let's illustrate the elimination method by solving the following system of equations:

$$80x_1 - 20x_2 - 20x_3 = 20 \quad \dots\dots\dots (1)$$

$$-20x_1 + 40x_2 - 20x_3 = 20 \quad \dots\dots\dots (2)$$

$$-20x_1 - 20x_2 + 130x_3 = 20 \quad \dots\dots\dots (3)$$

Solve Eq. (1) for x_1 . Thus,

$$x_1 = \frac{[20 - (-20)x_2 - (-20)x_3]}{80} \quad \dots\dots\dots (4)$$

Substituting Eq. (4) into Eq. (2) gives

$$-20 \left\{ \frac{[20 - (-20)x_2 - (-20)x_3]}{80} \right\} + 40x_2 - 20x_3 = 20 \quad \dots\dots\dots (5)$$

which can be simplified to give

$$35x_2 - 25x_3 = 25 \quad \dots\dots\dots (6)$$

Substituting Eq. (4) into Eq. (3) gives

$$-20\left\{\frac{[20 - (-20)x_2 - (-20)x_3]}{80}\right\} - 20x_2 + 130x_3 = 20 \quad \dots\dots\dots (7)$$

which can be simplified to give

$$-25x_2 + 125x_3 = 25 \quad \dots\dots\dots (8)$$

Next solve Eq. (6) for x_2 . Thus,

$$x_2 = \frac{[25 - (-25)x_3]}{35} \quad \dots\dots\dots (9)$$

Substituting Eq. (9) into Eq. (8) yields

$$-25\left\{\frac{[25 - (-25)x_3]}{35}\right\} + 125x_3 = 25 \quad \dots\dots\dots (10)$$

which can be simplified to give

$$\frac{750}{7}x_3 = \frac{300}{7} \quad \dots\dots\dots (11)$$

Thus, Eq's. (1, 2, and 3) has been reduced to the upper triangular system

$$80x_1 - 20x_2 - 20x_3 = 20 \quad \dots\dots\dots (12)$$

$$35x_2 - 25x_3 = 25 \quad \dots\dots\dots (13)$$

$$\frac{750}{7}x_3 = \frac{300}{7} \quad \dots\dots\dots (14)$$

which is equivalent to the original equation. This completes the elimination process.

Back substitution

The solution to Eq's. (12, 13, and 14) is accomplished easily by back substitution. Starting with equation (14) and working backward yields

$$x_3 = \frac{300}{750} = 0.40$$

$$x_2 = \frac{[25 - (-25)(0.40)]}{35} = 1.00$$

$$x_1 = \frac{[20 - (-20)(1.00) - (-20)(0.40)]}{80} = 0.60$$

Simple Elimination

The elimination procedure illustrated previously involves manipulation of the coefficient matrix A and the nonhomogeneous vector b . components of the x vector are fixed in their locations in the set of equations. As long as the columns are not interchanged, column j corresponds to x_j . Consequently, the x_j notation does not need to be carried throughout the operations. Only the numerical elements of A and b need to be considered. Thus, the elimination procedure can be simplified by augmenting the A matrix with the b vector and performing the row operations on the elements of the augmented A matrix to accomplish the elimination process, then performing the back substitution process to determine the solution vector. This simplified elimination procedure is illustrated in the following example.

Example: Solve the previous example using simple elimination?

Solution: The A matrix augmented by the b vector is

$$[A | b] = \left[\begin{array}{ccc|c} 80 & -20 & -20 & 20 \\ -20 & 40 & -20 & 20 \\ -20 & -20 & 130 & 20 \end{array} \right]$$

Performing the row operation to accomplish the elimination process yields:

$$\left[\begin{array}{ccc|c} 80 & -20 & -20 & 20 \\ -20 & 40 & -20 & 20 \\ -20 & -20 & 130 & 20 \end{array} \right] \begin{array}{l} \\ R_2 - (-20/80) R_1 \\ R_3 - (-20/80) R_1 \end{array}$$

$$\left[\begin{array}{ccc|c} 80 & -20 & -20 & 20 \\ 0 & 35 & -25 & 25 \\ 0 & -25 & 125 & 25 \end{array} \right] R_3 - (-25/35) R_2$$

$$\left[\begin{array}{ccc|c} 80 & -20 & -20 & 20 \\ 0 & 35 & -25 & 25 \\ 0 & 0 & \frac{750}{7} & \frac{300}{7} \end{array} \right] \Rightarrow \begin{aligned} x_1 &= \frac{[20 - (-20)(1.00) - (-20)(0.40)]}{80} = 0.60 \\ x_2 &= \frac{[25 - (-25)(0.40)]}{35} = 1.00 \\ x_3 &= \frac{300}{750} = 0.40 \end{aligned}$$



Gauss in 1803

Pivoting

The element on the major diagonal is called the **pivot** element. The elimination procedure described so far fails immediately if the first pivot element a_{11} is zero. The procedure also fails if any subsequent pivot element a_{ij} is zero. Even though there may be no zeros on the major diagonal in the original matrix, the elimination process may create zeros on the major diagonal. The simple elimination procedure described so far must be modified to avoid zeros on the major diagonal. This result can be accomplished by rearranging the equations, by interchanging equations (rows) or variables (columns), before each elimination step to put the element of largest magnitude on the diagonal. This process is called **pivoting**. Interchanging both rows and columns called **full pivoting**. Full pivoting is quite complicated, and thus it is rarely used. Interchanging only rows is called **partial pivoting**.

Pivoting eliminates zeros in the pivot element locations during the elimination process. Pivoting also reduces round-off errors, since the pivot element is a divisor during the elimination process, and division by large numbers introduces smaller round-off errors than division by small numbers. When the procedure is repeated, round-off errors can compound. This problem becomes more severe as the number of equations is increased.

Example: Elimination with pivoting to avoid zero pivot elements?

Solution: Using simple elimination with partial pivoting to solve the following system of linear algebraic equations, $Ax = b$:

$$\begin{bmatrix} 0 & 2 & 1 \\ 4 & 1 & -1 \\ -2 & 3 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ -3 \\ 5 \end{bmatrix}$$

Let's apply the elimination procedure by augmenting A with b . The first pivot element is zero, so pivoting is required. The largest number (in magnitude) in the first column under the pivot element occurs in the second row. Thus, interchanging the first and second rows and evaluating the elimination multipliers yields

$$\left[\begin{array}{ccc|c} 4 & 1 & -1 & -3 \\ 0 & 2 & 1 & 5 \\ -2 & 3 & -3 & 5 \end{array} \right] \quad \begin{array}{l} R_2 - (0/4)R_1 \\ R_3 - (-2/4)R_1 \end{array}$$

Performing the elimination operations yields

$$\left[\begin{array}{ccc|c} 4 & 1 & -1 & -3 \\ 0 & 2 & 1 & 5 \\ 0 & \frac{7}{2} & -\frac{7}{2} & \frac{7}{2} \end{array} \right]$$

Although the pivot element in the second row is not zero, it is not the largest element in the second column underneath the pivot element. Thus, pivoting is called for again. Note that pivoting is based only on the rows below the pivot element. The rows above the pivot element have already been through the elimination process. Using one of the rows above element would destroy the elimination already accomplished. Interchanging the second and third rows and evaluating the elimination multiplier yields

$$\left[\begin{array}{ccc|c} 4 & 1 & -1 & -3 \\ 0 & \frac{7}{2} & -\frac{7}{2} & \frac{7}{2} \\ 0 & 2 & 1 & 5 \end{array} \right] \quad R_3 - (4/7)R_2$$

Performing the elimination operation yields

$$\left[\begin{array}{ccc|c} 4 & 1 & -1 & -3 \\ 0 & \frac{7}{2} & -\frac{7}{2} & \frac{7}{2} \\ 0 & 0 & 3 & 3 \end{array} \right] \Rightarrow \begin{array}{l} x_1 = -1 \\ x_2 = 2 \\ x_3 = 1 \end{array}$$

Scaling

The elimination process described so far can incur significant round-off errors when the magnitudes of the pivot elements are smaller than the magnitudes of the other elements in the equations containing the pivot elements. In such cases, scaling is employed to select the pivot elements. After pivoting, elimination is applied to the original equations. Scaling is employed only to select the pivot elements.

Example: Elimination with scaled pivoting to reduce round-off errors?

Solution: Let's investigate the advantage of scaling by solving the following linear system:

$$\begin{bmatrix} 3 & 2 & 105 \\ 2 & -3 & 103 \\ 1 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 104 \\ 98 \\ 3 \end{bmatrix}$$

Which has the exact solution $x_1 = -1.0$, $x_2 = 1.0$, and $x_3 = 1.0$. To accentuate the effects of round-off, carry only three significant figures in the calculations. For the first column, pivoting does not appear to be required. Thus, the augmented A matrix and the first set of row operations are given by

$$\left[\begin{array}{ccc|c} 3 & 2 & 105 & 104 \\ 2 & -3 & 103 & 98 \\ 1 & 1 & 3 & 3 \end{array} \right] \quad \begin{array}{l} R_2 - (0.667) R_1 \\ R_3 - (0.333) R_1 \end{array}$$

which gives

$$\left[\begin{array}{ccc|c} 3 & 2 & 105 & 104 \\ 0 & -4.33 & 33.0 & 28.6 \\ 0 & 0.334 & -32.0 & -31.6 \end{array} \right] \quad R_3 - (-0.0771) R_2$$

Pivoting is not required for the second column. Performing the elimination indicated yields the triangularized matrix

$$\left[\begin{array}{ccc|c} 3 & 2 & 105 & 104 \\ 0 & -4.33 & 33.0 & 28.6 \\ 0 & 0 & -29.5 & -29.4 \end{array} \right] \Rightarrow \begin{array}{l} x_1 = -0.844 \\ x_2 = 0.924 \\ x_3 = 0.997 \end{array}$$

which does not agree very well with the exact solution

$$x_1 = -1.0$$

$$x_2 = 1.0$$

$$x_3 = 1.0$$

Round-off errors due to the three-digit precision have polluted the solution.

The effect of round-off can be reduced by scaling of equations before pivoting. Since scaling itself introduces round-off, it should be used only to determine if pivoting is required. All calculation should be made with the original unscaled equations.

Let's rework the problem using scaling to determine if pivoting is required. The first step in the elimination procedure eliminates all the elements in the first column under element a_{11} . Before performing that step, let's scale all the elements in column 1 by the largest element in each row. The result is

$$a_1 = \begin{bmatrix} 3/105 \\ 2/103 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 0.0286 \\ 0.0194 \\ 0.3333 \end{bmatrix}$$

Where the notation a_1 denotes the column vector consisting of the scaled elements from the first column of matrix A . The third element of a_1 is the largest element in a_1 , which indicates that rows 1 and 3 of matrix A should be interchanged. Thus, the previous equation with the elimination multipliers indicated, becomes

$$\left[\begin{array}{ccc|c} 1 & 1 & 3 & 3 \\ 2 & -3 & 103 & 98 \\ 3 & 2 & 105 & 104 \end{array} \right] \quad \begin{array}{l} R_2 - (2/1)R_1 \\ R_3 - (3/1)R_1 \end{array}$$

Performing the elimination and indicating the next elimination multiplier yields

$$\left[\begin{array}{ccc|c} 1 & 1 & 3 & 3 \\ 0 & -5 & 97 & 92 \\ 0 & -1 & 96 & 95 \end{array} \right] \quad R_3 - (1/5)R_2$$

Scaling the second and third elements of column 2 gives

$$a_2 = \begin{bmatrix} - \\ -5/97 \\ -1/96 \end{bmatrix} = \begin{bmatrix} - \\ -0.0516 \\ -0.0104 \end{bmatrix}$$

Consequently, pivoting is not indicated. Performing the elimination indicated yields

$$\left[\begin{array}{ccc|c} 1.0 & 1.0 & 3.0 & 3.0 \\ 0.0 & -5.0 & 97.0 & 92.0 \\ 0.0 & 0.0 & 76.6 & 76.6 \end{array} \right]$$

And performing back substitution yields $x_1 = -1.0$, $x_2 = 1.0$, and $x_3 = 1.0$, which is the exact solution.

Programming

The Gauss elimination procedure, in a format suitable for programming on a computer, is summarized as follows:

1. Define the $n \times n$ coefficient matrix A , the $n \times 1$ column vector b , and the $n \times 1$ order vector o .
2. Starting with column 1, scale column k ($k = 1, 2, \dots, n-1$) and search for the element of largest magnitude in column k and pivot (interchange rows) to put the coefficient into the $a_{k,k}$ pivot position. This step is accomplished by interchanging the corresponding elements of the $n \times 1$ order vector o .
3. For column k ($k = 1, 2, \dots, n-1$), apply the elimination procedure to rows i ($i = k+1, k+2, \dots, n$) to create zeros in column k below the pivot element, $a_{k,k}$. Do not actually calculate the zeros in column k . In fact, storing the elimination multipliers, $em = (a_{i,k}/a_{k,k})$, in place of the eliminated elements, $a_{i,k}$, creates the Doolittle LU factorization that will be presented later. Thus,

$$a_{i,j} = a_{i,j} - \left(\frac{a_{i,k}}{a_{k,k}} \right) a_{k,j} \quad (i, j = k+1, k+2, \dots, n)$$

$$b_i = b_i - \left(\frac{a_{i,k}}{a_{k,k}} \right) b_k \quad (i = k+1, k+2, \dots, n)$$

4. After step 3 is applied to all k columns, ($k = 1, 2, \dots, n-1$), the original A matrix is upper triangular.
5. 4. Solve for x using back substitution. If more than one b vector is present, solve for the corresponding x vectors one at a time. Thus,

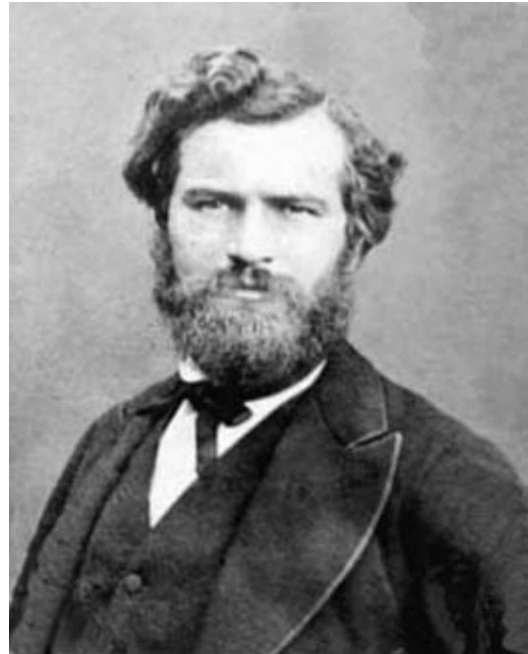
$$x_n = \frac{b_n}{a_{n,n}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{i,j} x_j}{a_{i,i}} \quad (i = n-1, n-2, \dots, 1)$$

Gauss-Jordan Elimination

Gauss-Jordan elimination is a variation of Gauss elimination in which the elements above the major diagonal are eliminated (made zero) as well as the elements below the major diagonal. The A matrix is transformed to a diagonal matrix. The rows are usually scaled to yield unity diagonal elements, which transforms the A matrix to the identity matrix, I . The transformed b vector is then the solution vector x .

The numbers of multiplications and divisions for Gauss-Jordan elimination is approximately $N = (n^3 / 2 - n / 2) + n^2$, which is approximately 50 percent larger than of Gauss elimination. Consequently, Gauss elimination is preferred.



Marie Ennemond Camille Jordan

1838 - 1922

Example: Gauss-Jordan elimination?

Solution: Let's rework the previous example using simple Gauss-Jordan elimination, that is, elimination without pivoting. The augmented A matrix is

$$\left[\begin{array}{ccc|c} 80 & -20 & -20 & 20 \\ -20 & 40 & -20 & 20 \\ -20 & -20 & 130 & 20 \end{array} \right] \quad R_1 / 80$$

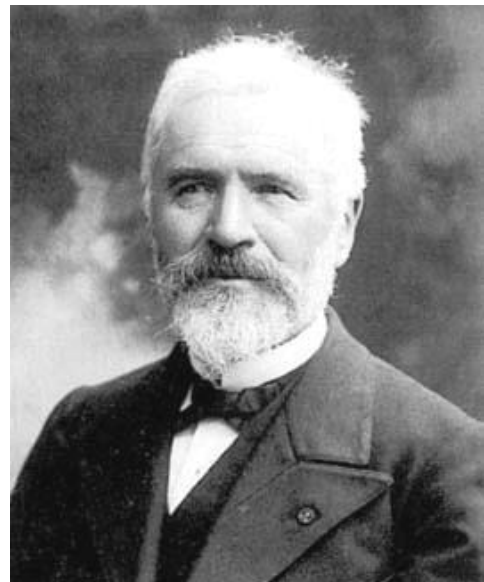
Scaling row 1 to give $a_{11} = 1$ gives

$$\left[\begin{array}{ccc|c} 1 & -1/4 & -1/4 & 1/4 \\ -20 & 40 & -20 & 20 \\ -20 & -20 & 130 & 20 \end{array} \right] \quad \begin{array}{l} R_2 - (-20)R_1 \\ R_3 - (-20)R_1 \end{array}$$

Applying elimination below row 1 yields

$$\left[\begin{array}{ccc|c} 1 & -1/4 & -1/4 & 1/4 \\ 0 & 35 & -25 & 25 \\ 0 & -25 & 125 & 25 \end{array} \right] \quad R_2 / 35$$

Scaling row 2 to give $a_{22} = 1$ gives



$$\left[\begin{array}{ccc|c} 1 & -1/4 & -1/4 & 1/4 \\ 0 & 1 & -5/7 & 5/7 \\ 0 & -25 & 125 & 25 \end{array} \right] \quad \begin{array}{l} R_1 - (-1/4)R_2 \\ R_3 - (-25)R_2 \end{array}$$

Applying elimination both above and below row 2 yields

$$\left[\begin{array}{ccc|c} 1 & 0 & -3/7 & 3/7 \\ 0 & 1 & -5/7 & 5/7 \\ 0 & 0 & 750/7 & 300/7 \end{array} \right] \quad R_3 / (750/7)$$

Scaling row 3 to give $a_{33} = 1$ gives

$$\left[\begin{array}{ccc|c} 1 & 0 & -3/7 & 3/7 \\ 0 & 1 & -5/7 & 5/7 \\ 0 & 0 & 1 & 215 \end{array} \right] \quad \begin{array}{l} R_1 - (-3/7)R_3 \\ R_2 - (-5/7)R_3 \end{array}$$

Applying elimination above row 3 completes the process.

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0.60 \\ 0 & 1 & 0 & 1.00 \\ 0 & 0 & 1 & 0.40 \end{array} \right]$$

The matrix A has been transformed to the identity matrix I and the b vector has been transformed to the solution vector, x . Thus, $x^T = [0.60 \ 1.00 \ 0.40]$.

The inverse of a square matrix A is the matrix A^{-1} such that $A \cdot A^{-1} = A^{-1} \cdot A = I$. Gauss-Jordan elimination can be used to evaluate the inverse of matrix A by augmenting A with the identity matrix I and applying the Gauss-Jordan algorithm. The transformed A matrix is the identity matrix I , and the transformed identity matrix is the matrix inverse, A^{-1} . Thus, applying Gauss-Jordan elimination yields

$$[A \mid I] \rightarrow [I \mid A^{-1}]$$

Example: Matrix inverse by Gauss-Jordan elimination?

Solution: Let's evaluate the inverse of matrix A presented in the last example. First, augment matrix A with the identity matrix, I . Thus,

$$[A \mid I] = \left[\begin{array}{ccc|ccc} 80 & -20 & -20 & 1 & 0 & 0 \\ -20 & 40 & -20 & 0 & 1 & 0 \\ -20 & -20 & 130 & 0 & 0 & 1 \end{array} \right]$$

Performing Gauss-Jordan elimination transforms the matrix to

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2/125 & 1/100 & 1/250 \\ 0 & 1 & 0 & 1/100 & 1/30 & 1/150 \\ 0 & 0 & 1 & 1/250 & 1/150 & 7/750 \end{array} \right]$$

from which

$$A^{-1} = \left[\begin{array}{ccc} 2/125 & 1/100 & 1/250 \\ 1/100 & 1/30 & 1/150 \\ 1/250 & 1/150 & 7/750 \end{array} \right] = \left[\begin{array}{ccc} 0.016000 & 0.010000 & 0.004000 \\ 0.010000 & 0.033333 & 0.007667 \\ 0.004000 & 0.007667 & 0.009333 \end{array} \right]$$

The Matrix Inverse Method

Systems of linear algebraic equations can be solved using the matrix inverse, A^{-1} . Consider the general system of linear algebraic equations:

$$Ax = b \quad \dots\dots\dots (1)$$

Multiplying Eq. (1) by A^{-1} yields

$$A^{-1}Ax = Ix = x = A^{-1}b$$

From which

$$x = A^{-1}b$$

Example: The matrix inverse method?

Solution: Let's solve the linear system considered in the previous example using the matrix inverse method.

$$A^{-1} = \begin{bmatrix} 2/125 & 1/100 & 1/250 \\ 1/100 & 1/30 & 1/150 \\ 1/250 & 1/150 & 7/750 \end{bmatrix} \quad b = \begin{bmatrix} 20 \\ 20 \\ 20 \end{bmatrix}$$

$$x = A^{-1}b = \begin{bmatrix} 2/125 & 1/100 & 1/250 \\ 1/100 & 1/30 & 1/150 \\ 1/250 & 1/150 & 7/750 \end{bmatrix} \begin{bmatrix} 20 \\ 20 \\ 20 \end{bmatrix}$$

$$x_1 = (2/125)(20) + (1/100)(20) + (1/250)(20) = 0.60$$

$$x_2 = (1/100)(20) + (1/30)(20) + (1/150)(20) = 1.00$$

$$x_3 = (1/250)(20) + (1/150)(20) + (7/750)(20) = 0.40$$

$$\text{Thus, } x^T = [0.60 \quad 1.00 \quad 0.40]$$

2. Iterative Methods

For many large systems of linear algebraic equations, $Ax = b$, the coefficient matrix A is extremely sparse. That is, most of the elements of A are zero. If the matrix is diagonally dominant,

$$|a_{i,i}| \geq \sum_{j=1, j \neq i}^n |a_{i,j}| \quad (i = 1, \dots, n) \text{ , with } > \text{ true for at least one row.}$$

it is generally more efficient to solve such systems of linear algebraic equations by iterative methods than by direct elimination methods.

Iterative methods begin by assuming an initial solution vector $x^{(0)}$. The initial solution vector is used to generate an improved solution vector $x^{(1)}$ based on some strategy for reducing the difference between $x^{(0)}$ and the actual solution vector x . This procedure is repeated (i.e., iterated) to convergence. The procedure is convergent if each iteration produces approximations to the solution vector that approach the exact solution vector as the number of iterations increases.

The number of iterations required to achieve convergence depends on:

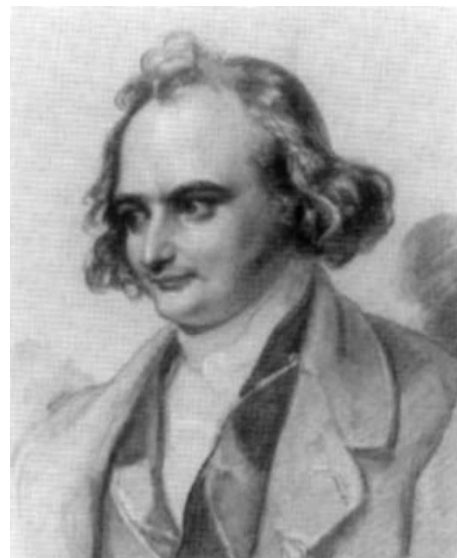
1. The dominance of the diagonal coefficients. As the diagonal dominance increases, the number of iterations required to satisfy the convergence criterion decreases.
2. The method of iteration used.
3. The initial solution vector.
4. The convergence criterion specified.

The Jacobi Iteration Method

Consider the general system of linear algebraic equations, $Ax=b$, written in index notation:

$$\sum_{j=1}^n a_{i,j} x_j = b_i \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (1)$$

In Jacobi iteration, each equation of the system is solved for the component of the solution vector associated with the diagonal element, that is, x_i . Thus,



Carl Gustav Jacob Jacobi

1804 - 1851

$$x_i = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j - \sum_{j=i+1}^n a_{i,j} x_j \right) \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (2)$$

An initial solution vector $\mathbf{x}^{(0)}$ is chosen. The superscript in parentheses denotes the iteration number, with zero denoting the initial solution vector. The initial solution vector $\mathbf{x}^{(0)}$ is substituted into Eq. (2) to yield the first improve solution vector $\mathbf{x}^{(1)}$. Thus,

$$x_i^{(1)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(0)} - \sum_{j=i+1}^n a_{i,j} x_j^{(0)} \right) \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (3)$$

This procedure is repeated (i.e., iterated) until some convergence criterion is satisfied. The Jacobi algorithm for the general iteration step (k) is:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} - \sum_{j=i+1}^n a_{i,j} x_j^{(k)} \right) \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (4)$$

An equivalent, but more convenient, form of Eq. (4) can be obtained by adding and subtracting $\mathbf{x}_i^{(k)}$ from the right-hand side of Eq. (4) to yield

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^n a_{i,j} x_j^{(k)} \right) \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (5)$$

Eq. (5) is generally written in the form

$$x_i^{(k+1)} = x_i^{(k)} + \frac{R_i^{(k)}}{a_{i,i}} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (6.a)$$

$$R_i^{(k)} = b_i - \sum_{j=1}^n a_{i,j} x_j^{(k)} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (6.b)$$

where the term $\mathbf{R}_i^{(k)}$ is called the **residual** of equation **i**. The residuals $\mathbf{R}_i^{(k)}$ are simply the net values of the equations evaluated for the approximate solution vector $\mathbf{x}^{(k)}$.

The Jacobi method is sometimes called the method of simultaneous iteration because all values of \mathbf{x}_i are iterated simultaneously. That is, all values of $\mathbf{x}_i^{(k+1)}$ depends only on the values of $\mathbf{x}_i^{(k)}$. The order of processing the equation is immaterial.

Example: The Jacobi Iteration Method.

Solution: To illustrate the Jacobi iteration method, let's solve the following system of linear algebraic equations:

$$\begin{bmatrix} 4 & -1 & 0 & 1 & 0 \\ -1 & 4 & -1 & 0 & 1 \\ 0 & -1 & 4 & -1 & 0 \\ 1 & 0 & -1 & 4 & -1 \\ 0 & 1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$$

which can be expanded to become

$$\begin{aligned} 4x_1 - x_2 + x_4 &= 100 \\ -x_1 + 4x_2 - x_3 + x_5 &= 100 \\ -x_2 + 4x_3 - x_4 &= 100 \\ x_1 - x_3 + 4x_4 - x_5 &= 100 \\ x_2 - x_4 + 4x_5 &= 100 \end{aligned}$$

These equations can be rearranged, using Eq. (6.b), to yield expressions for the residuals, R_i . Thus,

$$\begin{aligned} R_1 &= 100 - 4x_1 + x_2 - x_4 \\ R_2 &= 100 + x_1 - 4x_2 + x_3 - x_5 \\ R_3 &= 100 + x_2 - 4x_3 + x_4 \\ R_4 &= 100 - x_1 + x_3 - 4x_4 + x_5 \\ R_5 &= 100 - x_2 + x_4 - 4x_5 \end{aligned} \quad \dots \dots \dots (7)$$

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	25.000000	25.000000	25.000000	25.000000	25.000000
2	25.000000	31.250000	37.500000	31.250000	25.000000
3	25.000000	34.375000	40.625000	34.375000	25.000000
4	25.000000	35.156250	42.187500	35.156250	25.000000
5	25.000000	35.546875	42.578125	35.546875	25.000000
...
16	25.000000	35.714284	42.857140	35.714284	25.000000
17	25.000000	35.714285	42.857142	35.714285	25.000000
18	25.000000	35.714285	42.857143	35.714285	25.000000

To initiate the solution let $x^{(0)T} = [0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$. Substituting these values into Eq. (7) gives $R_i^{(0)} = 100.0 \ (i = 1, \dots, 5)$. Substituting these values into Eq. (6.a) gives $x_1^{(1)} = x_2^{(1)} = x_3^{(1)} = x_4^{(1)} = x_5^{(1)} = 25.0$. The procedure is then repeated with these values to obtain $x^{(2)}$, etc.

The first and subsequent iterations are summarized in the above table. Due to the symmetry of the coefficient matrix A and the symmetry of the b vector, $x_1 = x_5$ and $x_2 = x_4$. The calculations were carried out on a 13-digit precision computer and iterated until all $|\Delta x_i|$ changed by less than 0.000001 between iterations, which required 18 iterations.

Accuracy

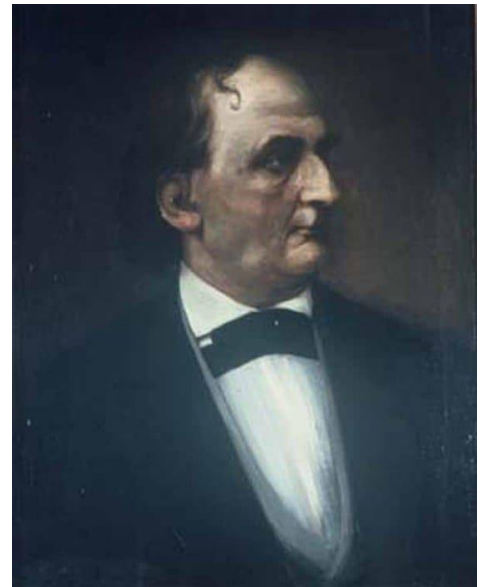
The accuracy of any approximate method is measured in terms of the error of the method. There are two ways to specify error: absolute error and relative error:

Absolute error = approximate value – exact value.

Relative error = absolute error/exact value.

The Gauss-Seidel Iteration Method

In the Jacobi method, all values of $x^{(k+1)}$ are based on $x^{(k)}$. The Gauss-Seidel method is similar to the Jacobi method, except that for most recently computed values of all x_i are used in all computations. In brief, as better values of x_i are obtained, use them immediately. Like the Jacobi method, the Gauss-Seidel method requires diagonal dominance to ensure convergence. The Gauss-Seidel algorithm is obtained from Jacobi algorithm, Eq. (4), by using $x_j^{(k+1)}$ values in the summation from $j=1$ to $i-1$ (assuming the sweeps through the equations proceed from $i=1$ to n). Thus,



Philipp Ludwig von Seidel

1821 - 1896

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i+1}^n a_{i,j} x_j^{(k)} \right) \quad (i = 1, 2, \dots, n) \quad \dots \dots \dots (8)$$

Illustration for 3×3 system of linear algebraic equations:

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2$$

$$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3$$

$$x_1^{(k+1)} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}} x_2^{(k)} - \frac{a_{13}}{a_{11}} x_3^{(k)}$$

$$x_2^{(k+1)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1^{(k+1)} - \frac{a_{23}}{a_{22}} x_3^{(k)}$$

$$x_3^{(k+1)} = \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}} x_1^{(k+1)} - \frac{a_{32}}{a_{33}} x_2^{(k+1)}$$

Equation (8) can be written in terms of the residuals R_i by adding and subtracting $x_i^{(k)}$ from the right-hand side of the equation and rearranging to yield

$$x_i^{(k+1)} = x_i^{(k)} + \frac{R_i^{(k)}}{a_{i,i}} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (9.a)$$

$$R_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i}^n a_{i,j} x_j^{(k)} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (9.b)$$

The Gauss-Seidel method is sometimes called the method of successive iteration because the most recent values of all x_i are used in all the calculations. Gauss-Seidel iteration generally converges faster than Jacobi iteration.

Example: The Gauss-Seidel Iteration Method.

Solution: let's rework the problem presented in the last example using Gauss-Seidel iteration. The residuals are given by Eq. (9.b). Substituting the initial solution vector, $x^{(0)T} = [0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$, into Eq. (9.b.1) gives $R_1^{(0)} = 100.0$. Substituting that result into Eq. (9.a.1) gives $x_1^{(1)} = 25.0$. Substituting $x^T = [25.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$ into Eq. (9.b.2) gives

$$R_2^{(0)} = (100.0 + 25.0) = 125.0$$

Substituting this result into Eq. (9.a.2) yields

$$x_2^{(1)} = 0.0 + \frac{125.0}{4} = 31.25$$

Continuing in this manner yields $R_3^{(0)} = 131.25$, $x_3^{(1)} = 32.81250$, $R_4^{(0)} = 107.81250$, $x_4^{(1)} = 26.953125$, $R_5^{(0)} = 95.703125$, $x_5^{(1)} = 23.925781$.

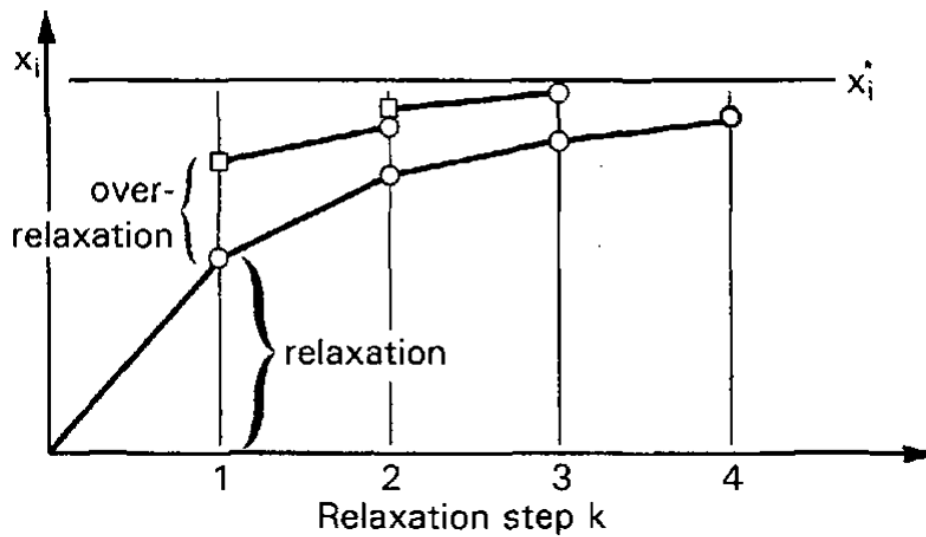
The first and subsequent iterations are summarized in the following table. The intermediate iterates are no longer symmetrical as they were in the last example. The calculations were carried out on a 13-digit precision computer and iterated until all $|\Delta x_i|$ changed by less than 0.000001 between iterations, which required 15 iterations, which is three less than required by the Jacobi method in the last example.

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	
1	25.000000	31.250000	32.812500	26.953125	23.925781
2	26.074219	33.740234	40.173340	34.506226	25.191498
3	24.808502	34.947586	42.363453	35.686612	25.184757
4	24.815243	35.498485	42.796274	35.791447	25.073240
5	24.926760	35.662448	42.863474	35.752489	25.022510
...
13	25.000002	35.714287	42.857142	35.714285	25.999999
14	25.000001	35.714286	42.857143	35.714285	25.000000
15	25.000000	35.714286	42.857143	35.714286	25.000000

The Successive-Over-Relaxation (SOR) Method

Iterative methods are frequently referred to as relaxation methods, since the iterative procedure can be viewed as relaxing $\mathbf{x}^{(0)}$ to the exact value \mathbf{x} . Historically, the method of relaxation, or just the term relaxation, refers to a specific procedure attributed to **Southwell** (1940).

Southwell observed that in many cases the change in x_i from iteration to iteration were always in the same directions. Consequently, over-correcting (i.e. over-relaxing) the values of x_i by the right amount accelerates convergence. This procedure is illustrated in the Figure below.



The Gauss-Seidel method can be modified to include over-relaxation simply by multiplying the residual $R_i^{(k)}$ in Eq. (9.a), by the over-relaxation factor, ω . Thus, the **successive-over-relaxation method** is given by

$$x_i^{(k+1)} = x_i^{(k)} + \omega \frac{R_i^{(k)}}{a_{i,i}} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (10.a)$$

$$R_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i}^n a_{i,j} x_j^{(k)} \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (10.b)$$

When $\omega=1.0$, Eq. (10.a) yields the Gauss-Seidel method. When $1.0 < \omega < 2.0$, the system of equations is over-relaxed. Over-relaxation is appropriate for systems of linear algebraic equations. when $\omega < 1.0$, the system of equations is under-relaxed. Under-relaxation is appropriate when the Gauss-Seidel algorithm causes the solution vector to overshoot and move farther away from the exact solution. This behavior is generally associated with the iterative solution of systems of nonlinear algebraic equations. The iterative method diverges

if $\omega \geq 2.0$. The relaxation factor does not change the final solution since it multiplies the residual R_i , which is zero when the final solution is reached.

The optimum value of the over-relaxation factor ω_{opt} depends on the size of the system of equations (i.e., the number of equations) and the nature of equations (i.e. the strength of the diagonal dominance, the structure of the coefficient matrix, etc.). As a general rule, larger values of ω_{opt} are associated with larger systems of equations.

Example: The SOR Method.

Solution: To illustrate the SOR method, let's rework the problem presented in the previous example using $\omega=1.1$. The residuals are given by Eq. (10.b). Substituting the initial value vector, $x^{(0)T} = [0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$, into Eq. (10.b.1) gives $R_1^{(0)}=100.0$. Substituting that value into Eq. (10.a.1) with $\omega=1.1$ gives

$$x_1^{(1)} = 0.0 + 1.10 \frac{100.0}{4} = 27.500000$$

Substituting $x^T = [27.5 \ 0.0 \ 0.0 \ 0.0 \ 0.0]$ into Eq. (10.b.2) gives

$$R_2^{(0)} = (100.0 + 27.5) = 127.5$$

Substituting this result into Eq. (10.a.2) yields

$$x_2^{(1)} = 0.0 + 1.10 \frac{127.5}{4} = 35.062500$$

Continuing in this manner yield the results presented in the next table.

The first and subsequent iterations are summarized in the following table. The calculations were carried out on a 13-digit precision computer and iterated until all $|\Delta x_i|$ changed by less than 0.000001 between iterations, which required 13 iterations, which is 5 less than required by the Jacobi method and 2 less than required by the Gauss-Seidel method in the last two examples, respectively. Its value becomes more significant as the numbers of equations increases.

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	27.500000	35.062500	37.142188	30.151602	26.149503
2	26.100497	34.194375	41.480925	35.905571	25.355629
3	24.419371	35.230346	42.914285	35.968342	25.167386
4	24.855114	35.692519	42.915308	35.790750	25.010375
5	24.987475	35.726188	42.875627	35.717992	24.996719
...
11	24.999996	35.714285	42.857145	35.714287	25.000000
12	25.000000	35.714286	42.857143	35.714286	25.000000
13	25.000000	35.714286	42.857143	35.714286	25.000000

The optimum value of ω can be determined by experimentation. If a problem is to be worked only once, that procedure is not worthwhile. However, if a problem is to be worked any times with the same A matrix for many different b vectors, then a search for ω_{opt} may be worthwhile. The following table presents the result of such a search for the problem considered in the last example. For this problem, $1.05 \leq \omega \leq 1.14$ yields the most efficient solution. Much more dramatic results are obtained for large systems of equations.

ω	k	ω	k	ω	k
1.00	15	1.06	13	1.12	13
1.01	14	1.07	13	1.13	13
1.02	14	1.08	13	1.14	13
1.03	14	1.09	13	1.15	14
1.04	14	1.10	13		
1.05	13	1.11	13		

Number of iterations as a function of ω

MATLAB APPLICATIONS

Simple Gauss

```
function [x, det] = gauss(A, b)
A = [1 3 4; 5 4 7; 0 2 3];
b = [1 2 4]';
% Solves A*x = b by Gauss elimination and computes det(A).
% USAGE: [x, det] = gauss(A, b)
if size(b, 2) > 1; b = b'; end % b must be column vector
n = length(b);
for k = 1:n-1 % Elimination phase
for i = k+1:n
if A(i, k) ~= 0
lambda = A(i, k) / A(k, k);
A(i, k+1:n) = A(i, k+1:n) - lambda * A(k, k+1:n);
b(i) = b(i) - lambda * b(k);
end
end
end
for k = n:-1:1 % Back substitution phase
b(k) = (b(k) - A(k, k+1:n) * b(k+1:n)) / A(k, k);
end
if nargin == 2; det = prod(diag(A)); end
for k = n:-1:1 % Back substitution phase
b(k) = (b(k) - A(k, k+1:n) * b(k+1:n)) / A(k, k);
end
x = b;
```

Gauss with Partial Pivoting

```
function x = gausspiv(A, B)
%The sizes of matrices A, B are supposed to be NA x NA and NA x NB.
%This function solves Ax = B by Gauss elimination algorithm with
pivoting.
clear; clc
A = [10 2 0 1; 5 1 3 1; 6 -3 0 1; 5 1 -1 -5];
B = [1 -2 -10 0]';
NA = size(A, 2); [NB1, NB] = size(B);
if NB1 ~= NA, error('A and B must have compatible dimensions');
end
N = NA + NB; AB = [A(1:NA, 1:NA) B(1:NA, 1:NB)] % Augmented matrix
epss = eps * ones(NA, 1);
for k = 1:NA
%Scaled Partial Pivoting at AB(k,k) by Eq.(2.2.20)
[akx, kx] = max(abs(AB(k:NA, k)) ./ ...
max(abs([AB(k:NA, k + 1:NA) epss(1:NA - k + 1)]'))));
if akx < eps, error('Singular matrix and No unique solution'); end
mx = k + kx - 1;
```

```

if kx > 1 % Row change if necessary
tmp_row = AB(k,k:N);
AB(k,k:N) = AB(mx,k:N);
AB(mx,k:N) = tmp_row;
end
AB
% Gauss forward elimination
AB(k,k + 1:N) = AB(k,k+1:N)/AB(k,k);
AB(k,k) = 1; %make each diagonal element one
for m = k + 1: NA
AB(m,k+1:N) = AB(m,k+1:N) - AB(m,k)*AB(k,k+1:N); %Eq. (2.2.5)
AB(m,k) = 0;
end
AB
end
%backward substitution for a upper-triangular matrix equation
% having all the diagonal elements equal to one
x(NA,:) = AB(NA,NA+1:N);
for m = NA-1: -1:1
x(m,:) = AB(m,NA + 1:N)-AB(m,m + 1:NA)*x(m + 1:NA,:); %Eq. (2.2.7)
end

```

Gauss-Seidel

```

function X = gauseid(A,B,X0,kmax)
%This function finds  $x = A^{-1} B$  by Gauss-Seidel iteration.
A=[0 2 0 1; 2 2 3 2; 4 -3 0 1; 6 1 -6 -5];
B=[0 -2 -7 6]';
%X0=[1 1 1 1];
%kmax=1000;
if nargin < 4, tol = 1e-6; kmax = 100;
elseif kmax < 1, tol = max(kmax,1e-16); kmax = 1000;
else tol = 1e-6;
end; if nargin < 4, tol = 1e-6; kmax = 100; end
if nargin < 3, X0 = zeros(size(B)); end
NA = size(A,1); X = X0;
for k = 1: kmax
X(1,:) = (B(1,:)-A(1,2:NA)*X(2:NA,:))/A(1,1);
for m = 2:NA-1
tmp = B(m,:)-A(m,1:m-1)*X(1:m - 1,:)-A(m,m + 1:NA)*X(m + 1:NA,:);
X(m,:) = tmp/A(m,m); %Eq. (2.5.4)
end
X(NA,:) = (B(NA,:)-A(NA,1:NA - 1)*X(1:NA - 1,:))/A(NA,NA);
if nargout == 0, X, end %To see the intermediate results
if norm(X - X0)/(norm(X0) + eps)<tol, break; end
X0 = X
end

```

```
% Gauss-Seidel Method in MATLAB

function x = gauss_siedel( A ,B )
disp ( 'Enter the system of linear equations in the form of AX=B')

%Inputting matrix A

A = input ( 'Enter matrix A :  \n')
% check if the entered matrix is a square matrix

[na , ma ] = size (A);
if na ~= ma
    disp('ERROR: Matrix A must be a square matrix')
    return
end

% Inputting matrix B

B = input ( 'Enter matrix B :  ')
% check if B is a column matrix

[nb , mb ] = size (B);
if nb ~= na || mb~=1
    disp( 'ERROR: Matrix B must be a column matrix')
    return
end

% Separation of matrix A into lower triangular and upper
triangular matrices
```

Numerical Analysis

DWE3214

PART-I: BASIC TOOLS

Unit-4: Numerical Differentiation & Integration



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-4: Numerical Differentiation and Integration

Numerical Differentiation

The evaluation of the derivatives is required in many problems in engineering and science:

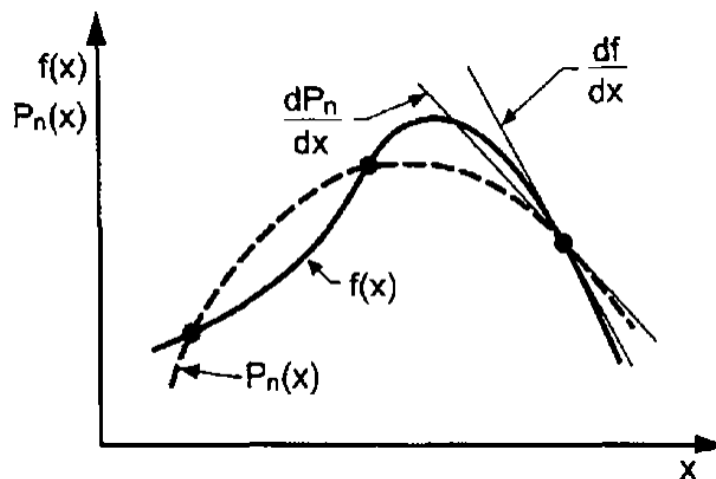
$$\frac{d}{dx}(f(x)) = f'(x) = f_x(x)$$

The function $f(x)$, which is to be differentiated, may be a known function or a set of discrete data. In general, known functions can be differentiated exactly. Differentiation of discrete data, however, requires an approximate numerical procedure.

Numerical differentiation formulas can be developed by fitting approximating functions (e.g., polynomials) to a set of discrete data and differentiating the approximating function. Thus,

$$\frac{d}{dx}(f(x)) \cong f'_x(x)$$

As illustrated in the figure below, even though the approximating polynomial $P_n(x)$ passes through the discrete data points exactly, the derivatives of the polynomial $P'_n(x)$ may not be a very accurate approximation of the derivative of the exact function $f(x)$ even at the known data points themselves. In general, numerical differentiation is an inherently inaccurate process.



The simple function

$$f(x) = \frac{1}{x}$$

which has the exact derivatives

$$\frac{d}{dx}\left(\frac{1}{x}\right) = f'(x) = -\frac{1}{x^2}$$

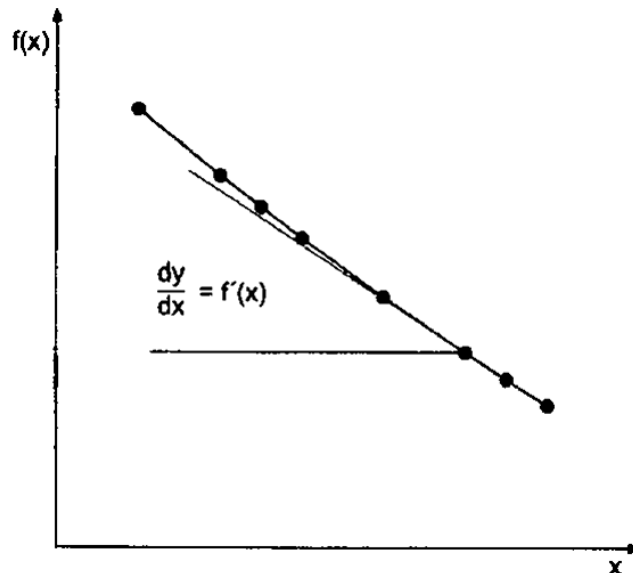
$$\frac{d^2}{dx^2}\left(\frac{1}{x}\right) = f''(x) = \frac{2}{x^3}$$

Can be considered to illustrate numerical differentiation procedures. In particular, at $x = 3.5$:

$$f'(3.5) = -\frac{1}{(3.5)^2} = -0.081633\dots$$

$$f''(3.5) = \frac{2}{(3.5)^3} = 0.046647\dots$$

x	f(x)
3.20	0.312500
3.30	0.303030
3.35	0.298507
3.40	0.294118
3.50	0.285714
3.60	0.277778
3.65	0.273973
3.70	0.270270



Direct Fit Polynomials

One of the most classical procedures of numerical differentiation is the direct fit polynomials, which can be used both for equally and unequally spaced sets of data. A direct fit polynomial procedure is based on fitting the data directly by a polynomial and differentiating the polynomial.

$$P_n(x) \cong a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

Where $P_n(x)$ is determined by one of the following methods:

1. Given $N = n+1$ points, $[x_i, f(x_i)]$, determine the exact n th-degree polynomial that passes through the data points, as will be discussed in unit 5.

2. Given $N > n+1$ points, $[x_i, f(x_i)]$, determine the least squares n th-degree polynomial that best fits the data points, as will be discussed in unit 5.

After approximating polynomial has been fit, the derivatives are determined by differentiating the approximating polynomial. Thus,

$$f'(x) \cong P'_n(x) = a_1 + 2a_2x + 3a_3x^2 + \dots$$

$$f''(x) \cong P''_n(x) = 2a_2 + 6a_3x + \dots$$

Example: Direct Fit Polynomials

Solution: Consider the following three data points:

x	$f(x)$
3.4	0.294118
3.5	0.285714
3.6	0.277778

First, fit the quadratic polynomial $P_2(x) = a_0 + a_1x + a_2x^2$, to the three data points:

$$0.294118 = a_0 + a_1(3.4) + a_2(3.4)^2$$

$$0.285714 = a_0 + a_1(3.5) + a_2(3.5)^2$$

$$0.277778 = a_0 + a_1(3.6) + a_2(3.6)^2$$

Solving for a_0 , a_1 , and a_2 by Gauss elimination gives

$$a_0 = 0.858314, \quad a_1 = -0.245500, \quad a_2 = 0.023400$$

$$\therefore P_2(x) = 0.858314 - 0.245500x + 0.023400x^2$$

$$\therefore P'_2(x) = -0.245500 + 0.04680x$$

$$P''_2(x) = 0.046800$$

Substituting $x = 3.5$ yields

$$P'_2(3.5) = -0.245500 + 0.04680(3.5) = -0.081700$$

Taylor Series Approach

This approach is especially useful for deriving finite difference approximations of exact derivatives (both total derivatives and partial derivatives) that appear in differential equations.

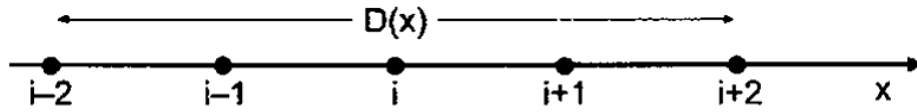
Difference formulas for function of a single variable, $f(x)$, can be developed from the Taylor series for a function of a single variable, Eq. (0.6):

$$f(x) = f_o + f_o' \Delta x + \frac{1}{2} f_o'' \Delta x^2 + \dots + \frac{1}{n!} f_o^{(n)} \Delta x^n + \dots$$

where $f_o = f(x_o)$, $f_o' = f'(x_o)$, etc. The continuous spatial domain $D(x)$ must be discretized into an equally spaced grid of discrete points, as illustrated in the figure below. For the discretized x space,

$$f(x_i) = f_i$$

where the subscript i denotes a particular spatial location. The Taylor series for $f(x)$ at grid points surrounding point i can be combined to obtain difference formulas for $f'(x_i)$, and $f''(x_i)$, etc.



Continuous spatial domain $D(x)$ and discretized x space.

The Taylor series for the function $f(x)$ can be rewritten as

$$f(x) = f_o + f_x|_o \Delta x + \frac{1}{2} f_{xx}|_o \Delta x^2 + \dots + \frac{1}{n!} f_{(n)x}|_o \Delta x^n + \dots$$

The Taylor formula with the remainder is given by Eq. (0.10):

$$f(x) = f_o + f_x|_o \Delta x + \frac{1}{2} f_{xx}|_o \Delta x^2 + \dots + \frac{1}{n!} f_{(n)x}|_o \Delta x^n + R^{n+1}$$

where the remainder term R^{n+1} is given by

$$R^{n+1} = \frac{1}{(n+1)!} f_{(n+1)x}(\xi) \Delta x^{n+1}$$

where $x_o \leq \xi \leq x_o + \Delta x$

Consider the equally spaced discrete finite difference grid illustrated in the above figure. Choose point i as the base point, and write the Taylor series for f_{i+1} and f_{i-1} :

$$f_{i+1} = f_i + f_x|_i \Delta x + \frac{1}{2} f_{xx}|_i \Delta x^2 + \frac{1}{6} f_{xxx}|_i \Delta x^3 + \frac{1}{24} f_{xxxx}|_i \Delta x^4 + \dots$$

$$f_{i-1} = f_i - f_x|_i \Delta x + \frac{1}{2} f_{xx}|_i \Delta x^2 - \frac{1}{6} f_{xxx}|_i \Delta x^3 + \frac{1}{24} f_{xxxx}|_i \Delta x^4 - \dots$$

Subtracting f_{i-1} from f_{i+1} gives

$$f_{i+1} - f_{i-1} = 2 f_x|_i \Delta x + \frac{1}{3} f_{xxx}|_i \Delta x^3 + \dots$$

Letting the f_{xxx} be the remainder term and solving for $f_x|_i$ yields

$$f_x|_i = \frac{f_{i+1} - f_{i-1}}{2 \Delta x} - \frac{1}{6} f_{xxx}(\xi) \Delta x^2$$

where $x_{i-1} \leq \xi \leq x_{i+1}$

The last equation is an exact expression for $f_x|_i$. If the remainder term is truncated it will yield an $O(\Delta x^2)$ finite difference approximation of $f_x|_i$. Thus,

$$f_x|_i = \frac{f_{i+1} - f_{i-1}}{2 \Delta x}$$

Adding f_{i-1} to f_{i+1} gives

$$f_{i+1} + f_{i-1} = 2 f_i + f_{xx}|_i \Delta x^2 + \frac{1}{12} f_{xxxx}|_i \Delta x^4 + \dots$$

Letting the f_{xxxx} term be the remainder term and solving for $f_{xx}|_i$ yields

$$f_{xx}|_i = \frac{f_{i+1} - 2 f_i + f_{i-1}}{\Delta x^2} - \frac{1}{12} f_{xxxx}(\xi) \Delta x^2$$

where $x_{i-1} \leq \xi \leq x_{i+1}$

Truncating the remainder term yields a finite difference approximation for $f_{xx}|_i$. Thus,

$$f_{xx}|_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

These difference equations are centered-difference formulas. They are inherently more accurate than one-sided formulas.

Example: Taylor series difference formulas.

Solution: Consider the following three data points:

x	$f(x)$
3.4	0.294118
3.5	0.285714
3.6	0.277778

$$f_x|_i = \frac{f_{i+1} - f_{i-1}}{2 \Delta x}$$

$$f'(3.5) = \frac{f(3.6) - f(3.4)}{2(0.1)} = \frac{0.277778 - 0.294118}{2(0.1)} = -0.08170$$

$$f_{xx}|_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

$$f''(3.5) = \frac{f(3.6) - 2f(3.5) + f(3.4)}{(0.1)^2} = \frac{0.277778 - 2(0.285714) + 0.294118}{(0.1)^2} = 0.0468$$

These are the same results that are obtained previously from the direct fit polynomial method

Error Estimation and Extrapolation

When the functional form of an error of a numerical algorithm is known, the error can be estimated by evaluating the algorithm for two different increment sizes. The error estimate can be used both for error control and extrapolation.

Consider a numerical algorithm which approximates an exact calculation with an error that depends on an increment, h . Thus,

$$f_{\text{exact}} = f(h) + Ah^n + Bh^{n+m} + Ch^{n+2m} + \dots \quad (1)$$

where n is the order of the leading error term and m is the increment in the order of the following error terms. Applying the algorithm at two increment sizes, $h_1 = h$ and $h_2 = h/R$, gives

$$f_{\text{exact}} = f(h) + Ah^n + O(h^{n+m}) \quad (2.a)$$

$$f_{\text{exact}} = f(h/R) + A(h/R)^n + O(h^{n+m}) \quad (2.b)$$

Subtracting Eq. (2.b) from Eq. (2.a) gives

$$0 = f(h) - f(h/R) + Ah^n - A(h/R)^n \quad (3)$$

Solving Eq. (3) for the leading error terms in Eqs. (2.a) and (2.b) yields

$$\text{Error}(h) = Ah^n = \frac{R^n}{R^n - 1} (f(h/R) - f(h)) \quad (4.a)$$

$$\text{Error}(h/R) = A(h/R)^n = \frac{1}{R^n - 1} (f(h/R) - f(h)) \quad (4.b)$$

Equation (4) can be used to estimate the leading error term in Eq. (2).

The error estimate can be added to the approximate results to yield an improved approximation. This process is called **extrapolation**. Adding Eq. (4.b) to Eq. (2.b) gives

$$\text{Extrapolated value} = f(h/R) + \frac{1}{R^n - 1} (f(h/R) - f(h)) + O(h^{n+m}) \quad (5)$$

The error of the extrapolated value is $O(h^{n+m})$. Two $O(h^{n+m})$ extrapolated results can be extrapolated to give an $O(h^{n+2m})$ result, where the exponent, n , in Eq. (5) is replaced with $n+m$. Higher-order extrapolations can be obtained by successive applications of Eq. (5)

Example: Error estimation and extrapolation.

Solution: Consider the following five data points:

x	$f(x)$
3.3	0.303030
3.4	0.294118
3.5	0.285714
3.6	0.277778
3.7	0.270270

In the last example we evaluated $f'(3.5)$ with $\Delta x = 0.1$. A more accurate result could be obtained by $f'(3.5)$ with $\Delta x = 0.05$, which requires data at $x = 3.45$ and 3.55 . While these points are not available, data are available at $x = 3.3$ and 3.7 , for which $\Delta x = 0.2$. Evaluating $f'(3.5)$ using $\Delta x = 0.2$ gives

$$f'(3.5) = \frac{f(3.7) - f(3.3)}{2(0.2)} = \frac{0.270270 - 0.303030}{2(0.2)} = -0.081900$$

The exact error in this result is $\text{Error} = -0.081900 - (-0.081633) = -0.000267$, which is approximately four times larger than the exact error obtained in the last example where $f'(3.5) = -0.081700$, for which the exact error is $\text{Error} = -0.081700 - (-0.081633) = -0.000067$.

Now that two estimates of $f'(3.5)$ are available, the error estimate for the result with the smaller Δx can be calculated from Eq. (4.b). Thus,

$$\text{Error}(h/R) = \frac{1}{R^n - 1} (f(h/R) - f(h))$$

$$\text{Error}(\Delta x/2) = \frac{1}{2^2 - 1} [-0.08177 - (-0.081900)] = 0.000067$$

Applying the extrapolation formula, Eq. (5), gives

$$\text{Extrapolated value} = f(h/R) + \frac{1}{R^n - 1} (f(h/R) - f(h)) + O(h^{n+m})$$

$$\text{Extrapolated value} = -0.081700 + 0.000067 = -0.081633$$

Which is the exact value to six digits after the decimal place.

Finite Difference Formulation

We can derive the same finite difference formulas using the general definition of the derivative, but without the ability to recognize the error order to establish the extrapolation process discussed earlier.

The general definition of the derivative can be illustrated as follows:

$$\frac{\Delta y}{\Delta x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

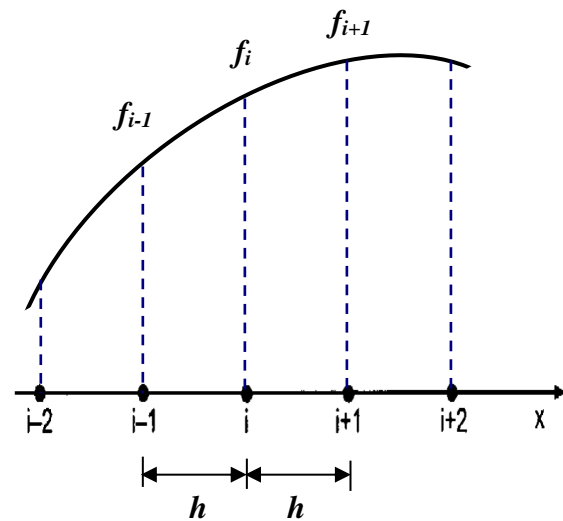
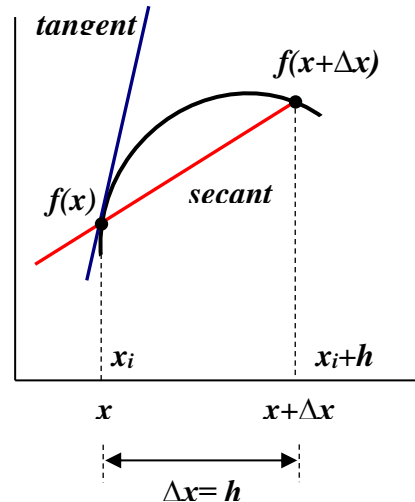
$$f_i^{(1)} = \frac{f_{i+1} - f_i}{h} \text{ Forward differentiation}$$

$$f_i^{(1)} = \frac{f_i - f_{i-1}}{h} \text{ Backward differentiation}$$

$$f_i^{(1)} = \frac{f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}}{h} \text{ Central differentiation}$$

OR

$$f_i^{(1)} = \frac{f_{i+1} - f_{i-1}}{2h}$$



$$f_i^{(1)} = \frac{1}{2h} [f_{i+1} - f_{i-1}]$$

$$f_i^{(2)} = \frac{1}{h} \left[f_{i+\frac{1}{2}}^{(1)} - f_{i-\frac{1}{2}}^{(1)} \right]$$

$$f_i^{(2)} = \frac{1}{h} \left[\frac{f_{i+1} - f_i}{h} - \frac{f_i - f_{i-1}}{h} \right]$$

$$f_i^{(2)} = \frac{1}{h} \left[\frac{f_{i+1} - f_i - f_i + f_{i-1}}{h} \right]$$

$$f_i^{(2)} = \frac{1}{h^2} [f_{i+1} - 2f_i + f_{i-1}]$$

$$f_i^{(3)} = \frac{1}{h^2} [f_{i+1}^{(1)} - 2f_i^{(1)} + f_{i-1}^{(1)}]$$

$$f_i^{(3)} = \frac{1}{h^2} \left[\frac{f_{i+2} - f_i}{2h} - 2 \left(\frac{f_{i+1} - f_{i-1}}{2h} \right) + \frac{f_i - f_{i-2}}{2h} \right]$$

$$f_i^{(3)} = \frac{1}{2h^3} [f_{i+2} - f_i - 2f_{i+1} + 2f_{i-1} + f_i - f_{i-2}]$$

$$f_i^{(3)} = \frac{1}{2h^3} [f_{i+2} - 2f_{i+1} + 2f_{i-1} - f_{i-2}]$$

$$f_i^{(4)} = \frac{1}{h^2} [f_{i+1}^{(2)} - 2f_i^{(2)} + f_{i-1}^{(2)}]$$

$$f_i^{(4)} = \frac{1}{h^2} \begin{bmatrix} f_{i-2} & f_{i-1} & f_i & f_{i+1} & f_{i+2} \\ \dots & \dots & \dots & \dots & \dots \\ & & 1 & -2 & 1 \\ & -2 & 4 & -2 & \\ 1 & -2 & 1 & & \end{bmatrix} \frac{1}{h^2}$$

$$f_i^{(4)} = \frac{1}{h^4} [f_{i-2} - 4f_{i-1} + 6f_i - 4f_{i+1} + f_{i+2}]$$

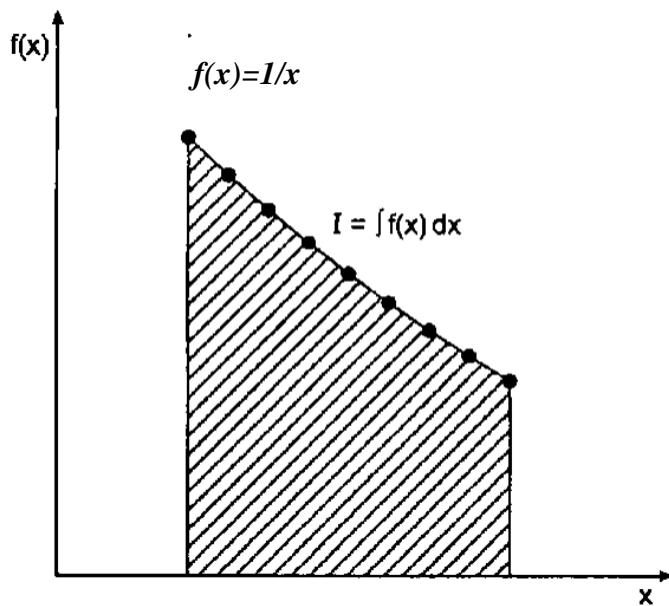
All the finite difference formula can be represented in **stencil** representation. Thus,

$$\begin{array}{l} f_i^{(1)} = \frac{1}{2h} \quad [\quad \times \quad \textcircled{-} \quad \textcircled{\textbf{0}} \quad \textcircled{1} \quad \times \quad] \\ f_i^{(2)} = \frac{1}{h^2} \quad [\quad \times \quad \textcircled{1} \quad \textcircled{\textbf{-}} \quad \textcircled{1} \quad \times \quad] \\ f_i^{(3)} = \frac{1}{2h^3} \quad [\quad \textcircled{-} \quad \textcircled{2} \quad \textcircled{\textbf{0}} \quad \textcircled{-} \quad \textcircled{1} \quad] \\ f_i^{(4)} = \frac{1}{h^4} \quad [\quad \textcircled{1} \quad \textcircled{-} \quad \textcircled{\textbf{6}} \quad \textcircled{-} \quad \textcircled{1} \quad] \end{array}$$

Numerical Integration

The evaluation of integrals, a process known as integration or quadrature, is required in many problems in engineering and science.

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



x	f(x)
3.1	0.32258065
3.2	0.31250000
3.3	0.30303030
3.4	0.29411765
3.5	0.28571429
3.6	0.27777778
3.7	0.27027027
3.8	0.26315789
3.9	0.25641026

The function $f(x)$, which is to be integrated, may be a known function or a set of discrete data. Some known functions have an exact integral, in which case the above equation $f(x) = 1/x$ can be evaluated exactly in closed form:

$$f(x) = \frac{1}{x}$$

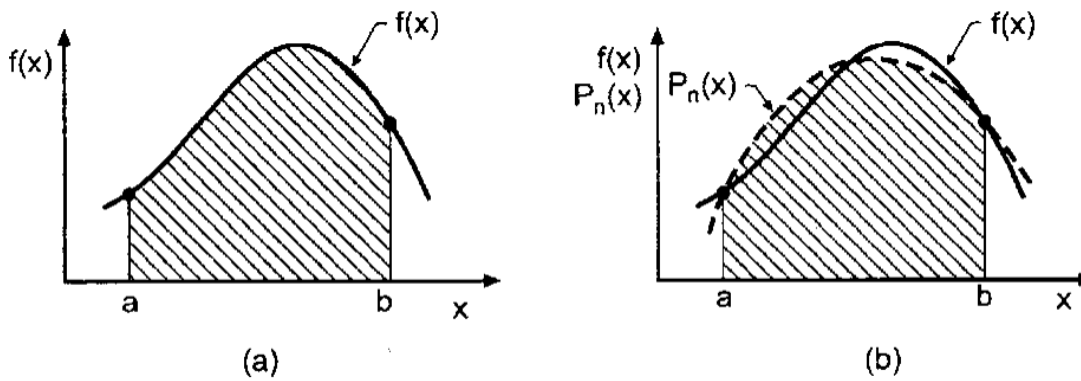
$$I = \int_{3.1}^{3.9} \frac{1}{x} dx = [\ln(x)]_{3.1}^{3.9} = \ln\left(\frac{3.9}{3.1}\right) = 0.22957444...$$

Many known functions, however, do not have an exact integral, and an approximate numerical procedure is required to evaluate $f(x)$. In many cases, the function $f(x)$ is known only at a set of discrete points, in which case an approximate numerical procedure is again required to evaluate $f(x)$.

Numerical integration (quadrature) formulas can be developed by fitting approximating functions (e.g., polynomials) to discrete data and integrating the approximating function:

$$I = \int_a^b f(x) dx \cong \int_a^b P_n(x) dx$$

This process is illustrated in the next figure.



Numerical integration. (a) Exact integral. (b) Approximate integral.

Direct Fit Polynomials

A straightforward numerical integration procedure that can be used for both unequally spaced data and equally spaced data is based on fitting the data by a direct fit polynomial and integrating that polynomial. Thus,

$$P_n(x) \cong a_0 + a_1x + a_2x^2 + \dots + a_nx^n + \dots$$

Where $P_n(x)$ is determined by one of the following methods:

1. Given $N = n+1$ points, $[x_i, f(x_i)]$, determine the exact n th-degree polynomial that passes through the data points, as discussed in chapter 3.
2. Given $N > n+1$ points, $[x_i, f(x_i)]$, determine the least squares n th-degree polynomial that best fits the data points, as discussed in chapter 3.
3. Given a known function $f(x)$ evaluate $f(x)$ at N discrete points and fit a polynomial by an exact fit or least squares fit.

After approximating polynomial has been fit, the integral becomes

$$I = \int_a^b f(x) dx \cong \int_a^b P_n(x) dx$$

$$I = \left(a_0 x + a_1 \frac{x^2}{2} + a_2 \frac{x^3}{3} + \dots \right)_a^b$$

Introducing the limits of integration and evaluation gives the value of the integral.

Example: Direct fit polynomial

Solution: Lets solve the example problem presented in the introduction by a direct fit polynomial. Recall:

$$I = \int_{3.1}^{3.9} \frac{1}{x} dx \cong \int_{3.1}^{3.9} P_n(x) dx$$

Consider the following three data points:

x	$f(x)$
3.1	0.32258065
3.5	0.28571429
3.9	0.25641026

Fit the quadratic polynomial, $P_2(x) \cong a_0 + a_1x + a_2x^2$, to the three data points by the direct fit method:

$$0.32258065 = a_0 + a_1(3.1) + a_2(3.1)^2$$

$$0.28571429 = a_0 + a_1(3.5) + a_2(3.5)^2$$

$$0.25641026 = a_0 + a_1(3.9) + a_2(3.9)^2$$

Solving for a_0 , a_1 , and a_2 by Gauss elimination gives

$$\therefore P_2(x) = 0.86470519 - 0.24813896x + 0.02363228x^2$$

$$I = \left[(0.86470519)x + \frac{1}{2}(-0.24813896)x^2 + \frac{1}{3}(0.02363228)x^3 \right]_{3.1}^{3.9}$$

$$I = 0.22957974$$

The error is Error = $0.02363228 - 0.22957444 = 0.00000530$

Newton-Cotes Formulas

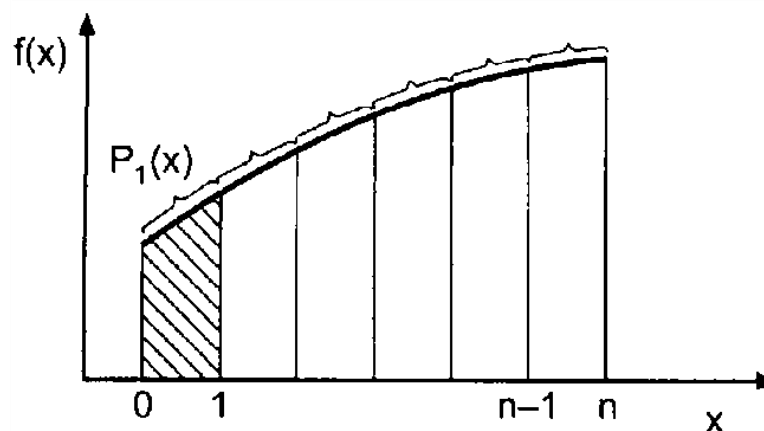
The direct fit polynomial procedure requires a significant amount of effort in the evaluation of the polynomial coefficients. When the function to be integrated is known at **equally spaced** points, the Newton forward-difference polynomial can be fit to the discrete data with much less effort. The resulting formulas are called **Newton-Cotes** formulas.

Each choice of the degree n of the interpolating polynomial yields a different Newton-Cotes formula. The next table the more common formulas. Higher-order formulas have been developed, but those presented in the table are sufficient for most problems in engineering and science. The rectangle rule has poor accuracy, so it is not considered further. The other three rules are developed in this section.

n	Formula
0	Rectangle rule
1	Trapezoid rule
2	Simpson's 1/3 rule
3	Simpson's 3/8 rule

The Trapezoid Rule

The trapezoid rule for single interval is obtained by fitting a first-degree polynomial to two discrete points, as illustrated in the figure below.



$$\Delta I = \frac{1}{2} h (f_o + f_1)$$

The composite trapezoid rule is obtained by applying ΔI over all the intervals of interest. Thus,

$$I = \sum_{i=0}^{n-1} \Delta I = \sum_{i=0}^{n-1} \frac{1}{2} h_i (f_i + f_{i+1})$$

where $h_i = x_{i+1} - x_i$. This equation does not require equally spaced data. When the data are equally spaced, the equation can be further simplified to:

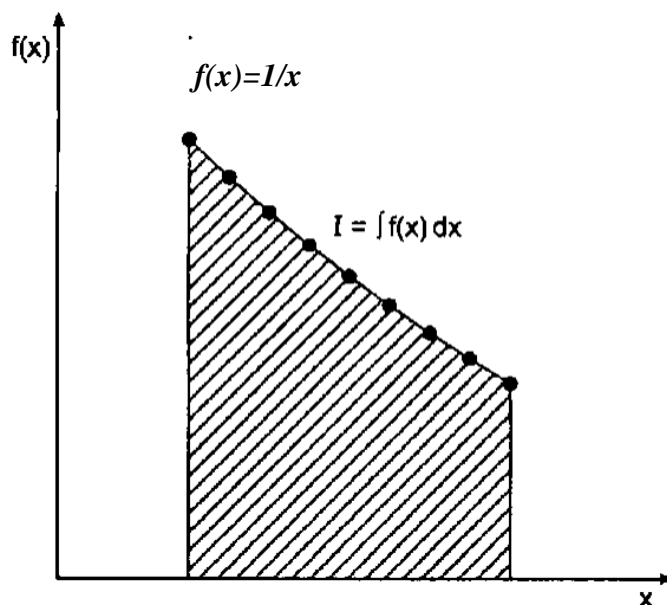
$$I = \frac{1}{2} h (f_o + 2 f_1 + 2 f_2 + \dots + 2 f_{n-1} + f_n)$$

where $\Delta x_i = \Delta x = h = \text{constant}$.

The Global Error of the Trapezoid rule is $O(h^2)$.

Example: The Trapezoid rule.

Solution: for the same function



x	f(x)
3.1	0.32258065
3.2	0.31250000
3.3	0.30303030
3.4	0.29411765
3.5	0.28571429
3.6	0.27777778
3.7	0.27027027
3.8	0.26315789
3.9	0.25641026

Solving the problem for the range of integration consisting of only one interval of $h=0.8$ gives

$$I(h=0.8) = \frac{0.8}{2} (0.32258065 + 0.25641026) = 0.23159636$$

Let's break the total range of integration into two intervals $h=0.4$ and apply the composite rule. Thus,

$$I(h=0.4) = \frac{0.4}{2} [0.32258065 + 2(0.28571429) + 0.25641026] = 0.23008389$$

For four intervals of $h=0.2$, the composite rule yields

$$I(h=0.2) = \frac{0.2}{2} [0.32258065 + 2(0.30303030 + 0.28571429 + 0.27027027) + 0.25641026] = 0.22970206$$

Finally, for eight intervals of $h=0.1$,

$$I(h=0.1) = \frac{0.1}{2} [0.32258065 + 2(0.31250000 + \dots + 0.26315789) + 0.25641026] = 0.22960636$$

Recall that the exact answer is $I=0.22967444$.

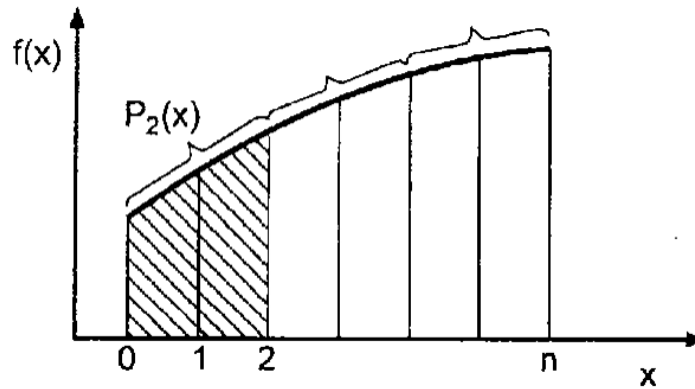
The results are tabulated in the following table, which also presents the errors and the ratios of the errors between successive interval halvings,

$$\text{Ratio} = \frac{E(h)}{E(h/2)} = \frac{O(h^2)}{O(h/2)^2} = 2^2 = 4$$

h	I	Error	Ratio
0.8	0.23159636	-0.00202192	
0.4	0.23008389	-0.00050945	3.97
0.2	0.22970206	-0.00012762	3.99
0.1	0.22960636	-0.00003192	4.00

Simpson's 1/3 Rule

Simpson's 1/3 rule is obtained by fitting a second-degree polynomial the three equally spaced discrete points, as illustrated in the figure below.



$$\Delta I = \frac{1}{3} h (f_0 + 4f_1 + f_2)$$

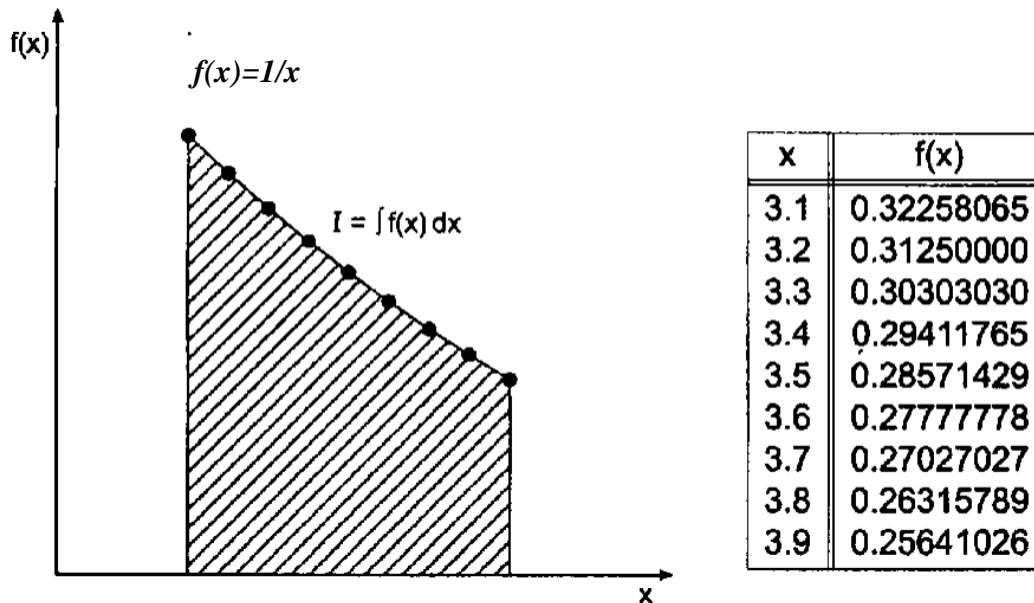
The composite Simpson's 1/3 rule for equally spaced points is obtained by applying ΔI over the entire range of integration. Note that the total number of increments must be even. Thus,

$$I = \frac{1}{3} h (f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \dots + 4f_{n-1} + f_n)$$

The Global Error of the Simpson's 1/3 rule is $O(h^4)$.

Example: Simpson's 1/3 rule.

Solution: for the same function



Solving the problem for two increments of $h=0.4$, the minimum permissible number of increments for Simpson's 1/3 rule, and one interval yields

$$I(h = 0.4) = \frac{0.4}{3} [0.32258065 + 4(0.28571429) + 0.25641026] = 0.22957974$$

Breaking the total range of integration into four increments of $h=0.2$ and two intervals and applying the composite rule yields,

$$I(h = 0.2) = \frac{0.2}{3} [0.32258065 + 4(0.30303030) + 2(0.28571429) + 4(0.27027027) + 0.25641026] = 0.22957478$$

Finally, for eight increments of $h=0.1$, and four intervals,

$$I(h = 0.1) = \frac{0.1}{3} [0.32258065 + 4(0.31250000) + 2(0.30303030) + 4(0.29411765) + 2(0.28571429) + 4(0.27777778) + 2(0.27027027) + 4(0.26315789) + 0.25641026] = 0.22957446$$

Recall that the exact answer is $I=0.22967444$.

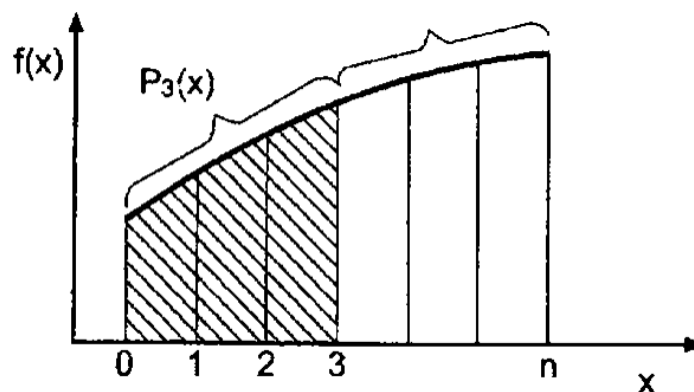
The results are tabulated in the following table, which also presents the errors and the ratios of the errors between successive increment sizes.

$$\text{Ratio} = \frac{E(h)}{E(h/2)} = \frac{O(h^4)}{O(h/2)^4} = 2^4 = 16$$

h	I	Error	Ratio
0.4	0.22957974	-0.00000530	15.59
0.2	0.22967478	-0.00000034	
0.1	0.22957446	-0.00000002	15.45

Simpson's 3/8 Rule

Simpson's 3/8 rule is obtained by fitting a third-degree polynomial the four equally spaced discrete points, as illustrated in the figure below.



$$\Delta I = \frac{3}{8} h (f_0 + 3f_1 + 3f_2 + f_3)$$

The composite Simpson's 3/8 rule for equally spaced points is obtained by applying ΔI over the entire range of integration. Note that the total number of increments must be a multiple of three. Thus,

$$I = \frac{3}{8} h (f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + \dots + 3f_{n-1} + f_n)$$

The Global Error of the Simpson's 1/3 rule is $O(h^4)$.

Simpson's 1/3 rule and Simpson's 3/8 have the same order, $O(h^4)$. As a matter of fact Simpson's 1/3 rule is practically more accurate than Simpson's 3/8 rule. In view of this result, what use, if any, is Simpson's 3/8 rule? Simpson's 3/8 rule is useful when the total number of increments is odd. Three increments can be evaluated by the 3/8 rule, and the remaining even number of increments can be evaluated by 1/3 rule.

Extrapolation and Romberg Integration (Error calculation)

When extrapolation is applied to numerical integration by the trapezoid rule, the result is called **Romberg integration**.

When the functional form of the error of a numerical algorithm is known, the error can be estimated by evaluating the algorithm for two different increment sizes. The error estimate can be used both for error control and extrapolation. Recall the error estimation formula, written for the process of numerical integration, that is with $f(h) = I(h)$. Thus,

$$\text{Error}(h/R) = \frac{1}{R^n - 1} (f(h/R) - f(h))$$

Where R is the ratio of the increment sizes and n is the global order of the algorithm. The extrapolation formula is given by

$$\text{Extrapolated value} = f(h/R) + \text{Error}(h/R)$$

Recall the composite trapezoid rule

$$I = \frac{1}{2} h (f_0 + 2 f_1 + 2 f_2 + \dots + 2 f_{n-1} + f_n)$$

It can be shown that the error of the composite trapezoid rule has the functional form

$$\text{Error} = C_1 h^2 + C_2 h^4 + C_3 h^6 + \dots$$

Thus, the basic algorithm is $O(h^2)$, so $n=2$. The following error terms increases in order of an increments of 2.

Let's apply the trapezoid rule for a succession of smaller and smaller increment sizes, where each successive increment size is one-half of the preceding increment size. Thus, $R = h/(h/2) = 2$. Applying the error estimation formula gives

$$\text{Error}(h/2) = \frac{1}{2^n - 1} [I(h/2) - I(h)]$$

For the trapezoid rule itself, $n=2$, and the last equation becomes

$$\text{Error}(h/2) = \frac{1}{3}[I(h/2) - I(h)]$$

This equation can be used for error estimation and error control.

Applying the extrapolation formula for $R=2$ gives

$$\text{Extrapolated value} = f(h/2) + \text{Error}(h/2) + O(h^4) + \dots$$

And this clearly illustrates that the result obtained by extrapolating the $O(h^2)$ trapezoid rule is $O(h^4)$.

If two extrapolated $O(h^4)$ values are available, which requires three $O(h^2)$ trapezoid rule result, those two values can be extrapolated to obtain an $O(h^6)$ value by applying the **Error(h/R)** equation with $n=4$ to estimate the $O(h^4)$ error, and adding that error to more accurate $O(h^4)$ values.

Example: Romberg integration.

Solution: Let's apply extrapolation to the results obtained earlier, in which the trapezoid rule is used.

Recall that the exact answer is $I=0.22967444$.

h	I
0.8	0.23159636
0.4	0.23008389
0.2	0.22970206
0.1	0.22960636

Substituting $I(h=0.8)$ and $I(h=0.4)$ yields

$$\text{Error}(h/2) = \frac{1}{3}[I(h/2) - I(h)]$$

$$\text{Error}(h/2) = \frac{1}{3}[0.23008389 - 0.23159636] = -0.00050416$$

$$\text{Extrapolated value} = f(h/2) + \text{Error}(h/2)$$

$$\text{Extrapolated value} = 0.23008389 + (-0.00050416) = 0.22957973$$

Repeating the procedure for the $h=0.4$ and $h=0.2$ results gives

$$\text{Error}(h/2) = \frac{1}{3}[I(h/2) - I(h)]$$

$$\text{Error}(h/2) = \frac{1}{3}[0.22970206 - 0.23008389] = -0.00012728$$

$$\text{Extrapolated value} = f(h/2) + \text{Error}(h/2)$$

$$\text{Extrapolated value} = 0.22970206 + (-0.00012728) = 0.22957478$$

Both extrapolated values are $O(h^4)$. Substituting the two $O(h^4)$ extrapolated values with $n=4$, gives an Error of

$$\text{Error}(h/2) = \frac{1}{2^n - 1}[I(h/2) - I(h)]$$

$$\text{Error}(h/2) = \frac{1}{2^4 - 1}[0.22957478 - 0.22957973] = -0.00000033$$

$$\text{Extrapolated value} = f(h/2) + \text{Error}(h/2)$$

$$\text{Extrapolated value} = 0.22957478 + (-0.00000033) = 0.22957445$$

These results, and the results of one more application of the trapezoid rule and its associated extrapolations, are presented in the next table.

The $O(h^4)$ results are identical to the results for the $O(h^4)$ Simpson's rule. The second $O(h^6)$ result agrees with the exact value to eight significant digits.

h	$I, O(h^2)$	Error	$O(h^4)$	Error	$O(h^6)$
0.8	0.23159636				
		-0.00050416	0.22957973		
0.4	0.23008389			-0.00000033	0.22957445
		-0.00012728	0.22957478		
0.2	0.22970206			-0.00000002	0.22957444
		-0.00003190	0.22957446		
0.1	0.22960636				

Multiple Integrals

The numerical integration formulas developed in the preceding sections for evaluating single integrals can be used to evaluate multiple integrals. Consider the double integral:

$$I = \int_c^d \int_a^b f(x, y) dx dy$$

This equation can be written in the form:

$$I = \int_c^d \left(\int_a^b f(x, y) dx \right) dy = \int_c^d F(y) dy$$

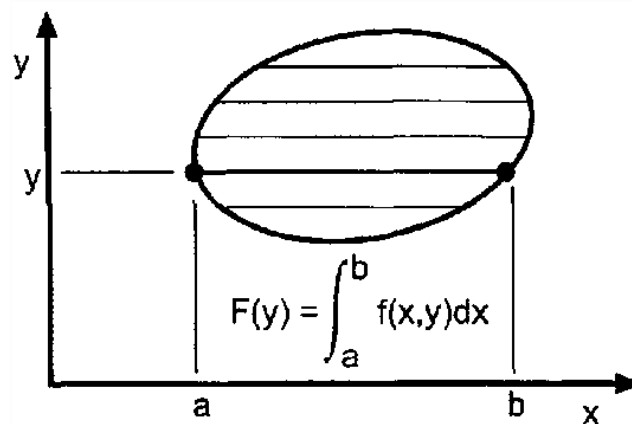
where

$$F(y) = \int_a^b f(x, y) dx \quad y = \text{constant}$$

The double integral is evaluated in two steps:

1. Evaluate $F(y)$ at selected values of y by any numerical integration formula.
2. Evaluate $I = \int F(y) dy$ by any numerical integration formula.

If the limits of integration are variable, as illustrated in the next figure, that must be accounted for.



Example: Double integral.

Evaluate the double integral $\int_{0.2}^{0.6} \int_{1.5}^{3.0} f(x, y) dx dy$ where $f(x, y)$ is given by the following table:

$y \backslash x$	0.1	0.2	0.3	0.4	0.5	0.6
0.5	0.165	0.428	0.687	0.942	1.190	1.431
1.0	0.271	0.640	1.003	1.359	1.703	2.035
1.5	0.447	0.990	1.524	2.045	2.549	3.031
2.0	0.738	1.568	2.384	3.177	3.943	4.672
2.5	1.216	2.520	3.800	5.044	6.241	7.379
3.0	2.005	4.090	6.136	8.122	10.030	11.841
3.5	3.306	6.679	9.986	13.196	16.277	19.198

Use Trapezoidal rule to integrate w.r.t. (x) and Simpson's 1/3 rule to integrate w.r.t. (y):

$$I_{y=0.2} = 0.2 : \int_{1.5}^{3.0} f(x, y) dx = \int_{1.5}^{3.0} f(x, 0.2) dx$$

$$I_{y=0.2} = \frac{h}{2} (f_1 + 2f_2 + 2f_3 + f_4)$$

$$I_{y=0.2} = \frac{0.5}{2} [0.990 + 2(1.568) + 2(2.520) + 4.090] = 3.3140$$

$$I_{y=0.3} = 0.3 : \int_{1.5}^{3.0} f(x, 0.3) dx = \int_{1.5}^{3.0} f(x, 0.3) dx$$

$$I_{y=0.3} = \frac{0.5}{2} [1.524 + 2(2.384) + 2(3.800) + 6.136] = 5.007$$

By the same way we get:

$$I_{y=0.4} = 6.6522 \quad I_{y=0.5} = 8.2368 \quad I_{y=0.6} = 9.7435$$

Now Accumulate these Integrations w.r.t. (y) using Simpson's 1/3 rule:

$$\int_{0.2}^{0.6} f(x, y) dy = \frac{0.1}{3} [3.314 + 4(5.007) + 2(6.6522) + 4(8.2368) + 9.7435] = 2.6446$$

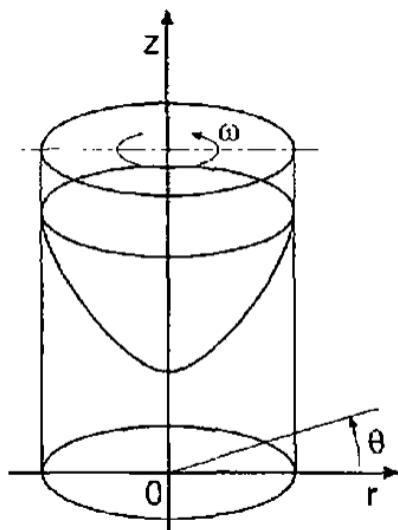
Example: Double integral.

Solution: to illustrate double integral with variable limits of integration, let's calculate the mass of water in a cylindrical container which is rotating at a constant angular velocity ω , as illustrated in the figure below, a meridional plane view through the axis of rotation is presented also. From a basic fluid mechanics analysis, it can be shown that the shape of the free surface is given by

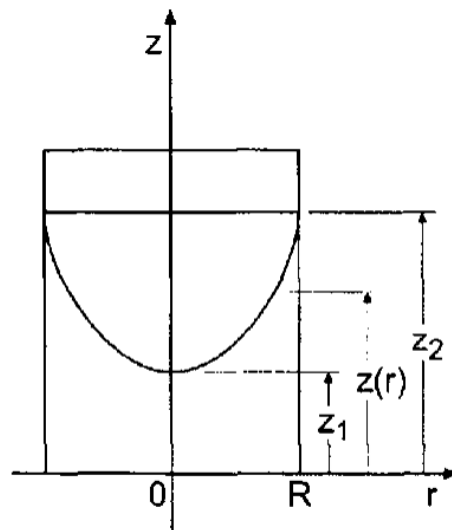
$$z(r) = A + B r^2 \quad \dots\dots\dots (1)$$

From measured data, $z(0)=z_1$ and $z(R)=z_2$. Substituting these values into Eq. (1) gives

$$z(r) = z_1 + \frac{(z_2 - z_1)}{R^2} r^2 \quad \dots\dots\dots (2)$$



(a) Physical arrangement.



(b) Meridional plane view.

In a specific experiment, $R=10.0 \text{ cm}$, $z_1=10.0 \text{ cm}$, and $z_2=20.0 \text{ cm}$. In this case Eq. (2) gives

$$z(r) = 10 + 0.1 r^2 \quad \dots\dots\dots (3)$$

Let's calculate the mass of the water in the container at this condition. The density of water is $\rho = 1.0 \text{ g/cm}^3$. Due to axial symmetry in the geometry of the container and the height distribution, the mass in the container can be expressed in cylindrical coordinates as

$$m = \int dm = \int_0^R \rho z(r) (2\pi r dr) = 2\pi \rho \int_0^R \left[z_1 + \frac{(z_2 - z_1)}{R^2} r^2 \right] \quad \dots\dots\dots (4)$$

which has the exact integral

$$m = \pi \rho \left[z_1 R^2 + \frac{(z_2 - z_1)}{2} R^2 \right] \dots\dots\dots (5)$$

Substituting the specified values of ρ , R , z_1 , and z_2 into Eq. (5) yields

$$m = 1500 \pi \text{ grams} = 4712.388980 \text{ g}$$

To illustrate the process of numerical double integration, let's solve this problem in Cartesian coordinates. The figure below illustrates a discretized Cartesian grid on the bottom of the container. In Cartesian coordinates, $dA = dx dy$, and the differential mass in a differential column of height $z(r)$ is given by

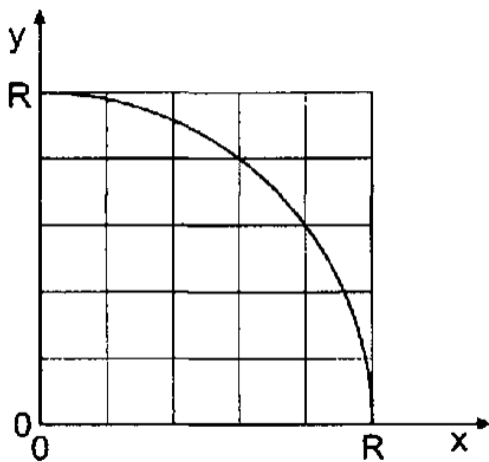
$$dm = \rho z(r) dA \dots\dots\dots (6)$$

Substituting Eq. (2) into Eq. (6), where $r^2 = x^2 + y^2$, and integrating gives

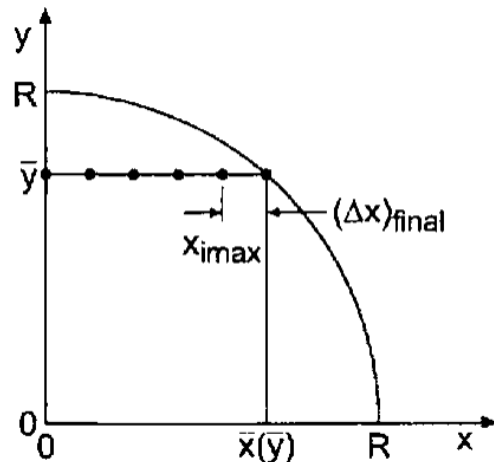
$$m = \int dm = \rho \iint \left[z_1 + (z_2 - z_1)(x^2 + y^2) R^{-2} \right] dx dy \dots\dots\dots (7)$$

Substituting the specific values of ρ , R , z_1 , and z_2 into Eq. (7) gives

$$m = 1.0 \iint \left[10 + 0.1(x^2 + y^2) \right] dx dy \dots\dots\dots (8)$$



(a) Discretized grid.



(b) Relationship of \bar{x} and \bar{y} .

Due to symmetry about the x and y axis, Eq. (8) can be expressed as

$$m = 4(1.0) \int_0^R \left\{ \int_0^{\bar{x}(\bar{y})} (10 + 0.1(x^2 + \bar{y}^2)) dx \right\} d\bar{y} = 4 \int_0^R F(\bar{y}) d\bar{y} \quad \dots\dots\dots (9)$$

where $F(\bar{y})$ is defined as

$$F(\bar{y}) = \int_0^{\bar{x}(\bar{y})} [10 + 0.1(x^2 + \bar{y}^2)] dx \quad \dots\dots\dots (10)$$

\bar{y} and $\bar{x}(\bar{y})$ are illustrated in the above figure. Thus,

$$\bar{x} = (R^2 - \bar{y}^2)^{1/2} \quad \dots\dots\dots (11)$$

Let's discretize the y -axis into **10** equally spaced increments with $\Delta y = 1.0$ cm. For each value of $\bar{y} = (j-1)\Delta y$ ($j=1,2,\dots,11$) let's calculate $\bar{x}(\bar{y})$ from Eq. (11). At each value of \bar{y} , let's discretize the x -axis into $(i \max(\bar{y})-1)$ equally spaced increments with $\Delta x = 1.0$ cm, with a final increment $(\Delta x)_{\text{final}}$ between $x = (i \max(\bar{y})-1)\Delta x$ and $\bar{x}(\bar{y})$. The resulting geometrical parameters are presented in the following table.

j	\bar{y} , cm	$\bar{x}(\bar{y})$, cm	$i \max(\bar{y})$	$x(i \max(\bar{y}))$	$(\Delta x)_{\text{final}}$
1	0.0	10.000000	11	10.0	0.000000
2	1.0	9.949874	10	9.0	0.949874
3	2.0	9.797959	10	9.0	0.797959
4	3.0	9.539392	10	9.0	0.539392
5	4.0	9.165151	10	9.0	0.165151
6	5.0	8.660254	9	8.0	0.660254
7	6.0	8.000000	9	8.0	0.000000
8	7.0	7.141428	8	7.0	0.141428
9	8.0	6.000000	7	6.0	0.000000
10	9.0	4.358899	5	4.0	0.358899
11	10.0	0.000000	1	0.0	0.000000

The values of $F(\bar{y})$, defined in Eq. (10), are evaluated by the trapezoid rule. As an example, consider $j=6$ for which $\bar{y} = 5.0$ cm. Equation (10) becomes

$$F(5.0) = \int_0^{8.00000} [10 + 0.1(x^2 + 25)] dx = \int_0^{8.00000} F(x) dx \quad \dots\dots\dots (12)$$

The following table presents the integrand $F(x)$ of Eq. (12).

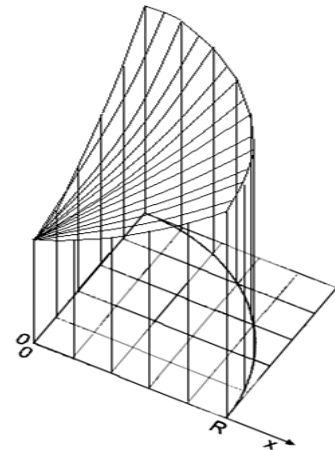
Integrating Eq. (12) by trapezoid rule gives

$$F(5.0) = \frac{1.0}{2} [12.500 + 2(12.600 + 12.900 + 13.400 + 14.000 + 15.000 + 16.100 + 17.400) + 18.9] + \frac{0.660254}{2} (18.900 + 20.000) = 130.041941$$

$\bar{y} = 5.0 \text{ cm}$			
i	$F(x)$	i	$F(x)$
1	12.500	6	15.000
2	12.600	7	16.100
3	12.900	8	17.400
4	13.400	9	18.900
5	14.100	10	20.000

Repeating this procedure for every value of \bar{y} in the geometrical parameters table yields the results presented in the table below.

Values of $F(\bar{y})$					
j	$\bar{y}, \text{ cm}$	$F(\bar{y})$	j	$\bar{y}, \text{ cm}$	$F(\bar{y})$
1	0.0	133.500000	7	6.0	126.000000
2	1.0	133.492600	8	7.0	118.664426
3	2.0	133.410710	9	8.0	105.700000
4	3.0	133.068144	10	9.0	81.724144
5	4.0	132.128255	11	10.0	0.000000
6	5.0	130.041941			



Integrating Eq. (9) by the trapezoid rule, using the values of $F(\bar{y})$ yields

$$m = 4.0 \frac{1.0}{2} [133.500 + 2(133.492600 + 133.410710 + 133.068144 + 132.128255 + 130.041941 + 126.000000 + 118.664426 + 105.700000 + 81.724144) + 0.000000] = 4643.920883 \text{ g}$$

The error is **Error=4643.920883-4712.388980=-68.468047 g.**

Numerical Analysis

DWE3214

PART-I: BASIC TOOLS

Unit-5: Interpolation and Curve Fitting



Dr. Zaid Al-Azzawi

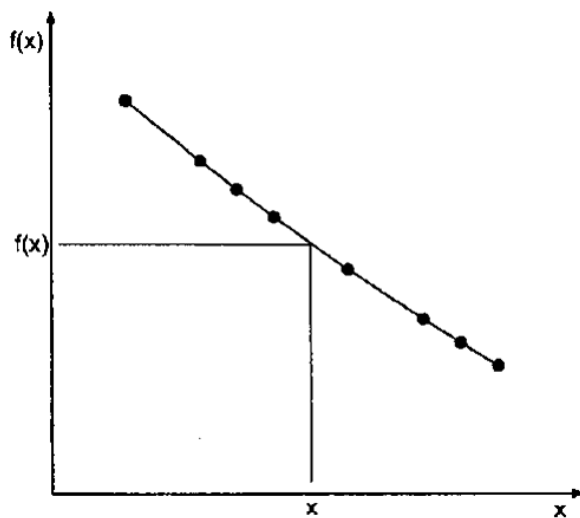
**University of Al-Anbar
College of Engineering**

2022/2023

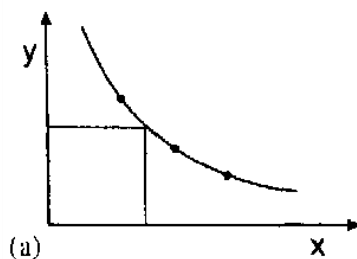
Unit-5: Interpolation and Curve Fitting

The Figure below illustrates a set of tabular data in the form of a set of $[x, f(x)]$ pairs. The function $f(x)$ is known at a finite set (actually eight) of discrete values of x . The value of the function can be determined at any of the eight values of x simply by a table lookup. However, a problem arises when the value of the function is needed at any value of x between the discrete values in the table. The actual function is not known and cannot be determined from the tabular values. However, the actual function can be approximated by some known function, and the value of the approximating function can be determined at any desired value of x . This process, which is called **interpolation**, is the subject of this Chapter. The discrete data of the figure below are actually values of the function $f(x) = 1/x$.

In many applications, the values of the discrete data at the specific points are not all that is needed. Values of the function at points other than the known discrete points may be needed (i.e., **interpolation**). The derivative of the function may be required (i.e., **differentiation**). The integral of the function may be of interest (i.e., **integration**). These processes are performed by **fitting** an approximating function to the set of discrete data and performing the desired process on the approximating function.

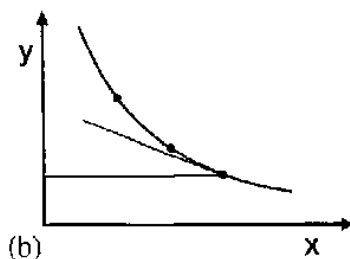


x	$f(x)$
3.20	0.312500
3.30	0.303030
3.35	0.298507
3.40	0.294118
3.50	0.285714
3.60	0.277778
3.65	0.273973
3.70	0.270270



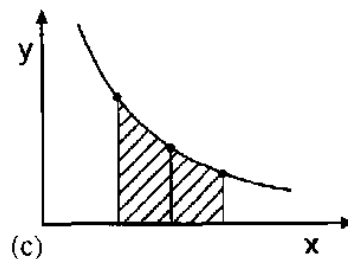
(a)

a) interpolation



(b)

b) differentiation



(c)

c) integration

Many types of approximating functions exist. In fact, any analytical function can be used as an approximating function. Three of the more common approximating functions are:

1. Polynomials.
2. Trigonometric functions.
3. Exponential functions.

Approximating functions should have the following properties:

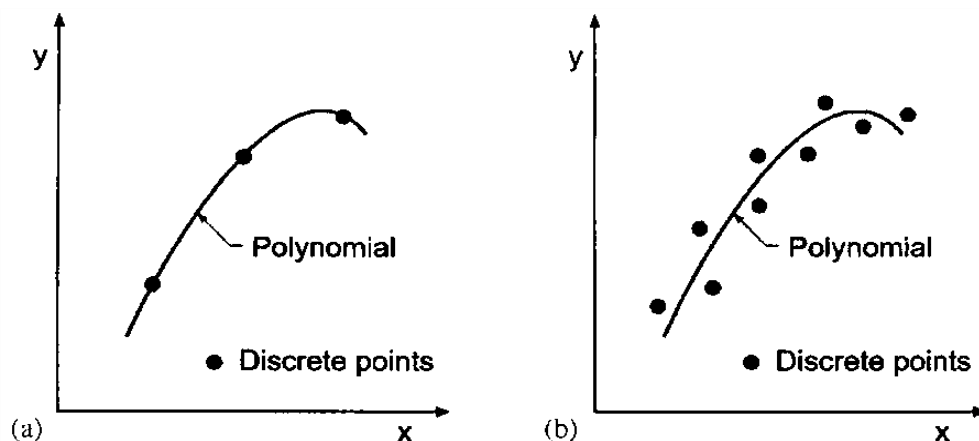
1. The approximating function should be easy to determine.
2. It should be easy to evaluate.
3. It should be easy to differentiate.
4. It should be easy to integrate.

Polynomials satisfy all four of these properties. Consequently, polynomial approximating functions are used here to fit sets of discrete data for interpolation, differentiation, and integration.

There are two fundamentally different ways to fit a polynomial to a set of discrete data:

1. Exact fits.
2. Approximate fits.

An **exact fit** yields a polynomial that passes exactly through all of the discrete points, as illustrated in the figure below. This type of fit is useful for small sets of smooth data. An **approximate fit** yields a polynomial that passes through the set of data in the best manner possible, without being required to pass exactly through any of the data points. Approximate fits are useful for large sets of smooth data and small or large sets of rough data.



Polynomial approximation. (a) Exact fit. (b) Approximate fit.

Direct Fit Polynomials

First let's consider a completely general procedure for fitting a polynomial to a set of equally spaced or unequally spaced data. Given $n+1$ sets of data $[x_0, f(x_0)], [x_1, f(x_1)], \dots, [x_n, f(x_n)]$, which will be written as $(x_0, f_0), (x_1, f_1), \dots, (x_n, f_n)$, determine the unique n th-degree polynomial $P_n(x)$ that passes exactly through the $n+1$ points:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \quad (1)$$

For simplicity of notation, let $f(x_i) = f_i$. Substituting each data point into Eq. (1) yields $n+1$ equations:

$$f_0 = a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n \quad (2.0)$$

$$f_1 = a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n \quad (2.1)$$

.....

$$f_n = a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n \quad (2.n)$$

There are $n+1$ linear equations containing the $n+1$ coefficients a_0 to a_n . Equation (2) can be solved for a_0 to a_n by Gauss elimination. The resulting polynomial is the unique n th-degree polynomial that passes exactly through the $n+1$ data points. The direct fit polynomial procedure work for both equally spaced data and unequally spaced data.

Example: Direct fit polynomials.

Solution: To illustrate interpolation by a direct fit polynomial, consider the simple function $y = f(x) = 1/x$, and construct the following set of six significant figure data:

x	$f(x)$
3.35	0.298507
3.40	0.294118
3.50	0.285714
3.60	0.277778

Let's interpolate for y at $x = 3.44$ using linear, quadratic, and cubic interpolation. The exact value is

$$y(3.44) = f(3.44) = \frac{1}{3.44} = 0.290698\dots$$

Let's illustrate the procedure in detail for a quadratic polynomial:

$$P_2(x) = a + b x + c x^2$$

To center data around $x = 3.44$, the first three points are used. Applying $P_2(x)$ at each of these data points gives the following three equations:

$$0.298507 = a + b(3.35) + c(3.35)^2$$

$$0.294118 = a + b(3.40) + c(3.40)^2$$

$$0.285714 = a + b(3.50) + c(3.50)^2$$

Solving these three equations for a , b , and c by Gauss elimination without scaling or pivoting yields

$$P_2(x) = 0.876561 - 0.256080x + 0.0249333x^2$$

Substituting $x = 3.44$ in the polynomial gives

$$P_2(3.44) = 0.876561 - 0.256080(3.44) + 0.0249333(3.44)^2 = 0.290697$$

The error is $\text{Error}(3.44) = P_2(3.44) - f(3.44) = 0.290697 - 0.290698 = -0.000001$.

For a linear polynomial, use $x = 3.4$ and 3.5 to center that data around $x = 3.44$. The resulting linear polynomial is

$$P_1(x) = 0.579854 - 0.0840400x$$

Substituting $x = 3.44$ in the polynomial gives $P_1(3.44) = 0.290756$

For cubic polynomial, all four points must be used. The resulting cubic polynomial is

$$P_3(x) = 1.121066 - 0.470839x + 0.0878000x^2 - 0.00613333x^3$$

Substituting $x = 3.44$ in the polynomial gives $P_3(3.44) = 0.290698$.

The results are summarized below, where the results of linear, quadratic, and cubic interpolation and the errors, $\text{Error}(3.44) = p_n(3.44) - 0.290698$, are tabulated. The advantages of higher-degree interpolation are obvious.

$P(3.44) = 0.290756$	Linear interpolation	Error = 0.000058
$= 0.290697$	Quadratic interpolation	$= -0.000001$
$= 0.290698$	Cubic interpolation	$= 0.000000$

The main advantage of direct fit polynomials is that the explicit form of the approximating function is obtained, and the interpolation at several values of \mathbf{x} can be accomplished simply by evaluating the polynomial at each value of \mathbf{x} . The work required to obtain the polynomial does not have to be redone for each value of \mathbf{x} . A second advantage is that the data can be unequally spaced.

The main disadvantage of direct fit polynomials is that each time the degree of the polynomial is changed, all of the work required to fit the new polynomial must be redone.

Direct Multivariate Polynomial Approximation

Many problems arise in engineering and science when the dependent variable is a function of two or more independent variables, for example, $z = f(\mathbf{x}, \mathbf{y})$ is a two variable or bivariate, function. Such functions in general called multivariate functions. When multivariate function is given by tabular data, multivariate approximation is required for interpolation, differentiation, and integration.

Consider the bivariate function, $z = f(\mathbf{x}, \mathbf{y})$, and the set of tabular data presented in the following table. The tabular data can be fit by a multivariate polynomial of the form

$$z = f(x, y) = a + bx + cy + dxy + ex^2 + fy^2 + gx^2y + hxy^2 + ix^3 + jy^3 + \dots$$

The number of data points must equal the number of coefficients in the polynomial. A linear bivariate polynomial in \mathbf{x} and \mathbf{y} is obtained by including the first four terms in the equation. A quadratic bivariate polynomial in \mathbf{x} and \mathbf{y} is obtained by including the first eight terms in the equation. The number of terms in approximating polynomial increases rapidly as the degree of approximations increases. This leads to ill-conditioned systems of linear equations for determining the coefficients. Consequently, multivariate high-degree approximation must be used with caution.

Example: Direct multivariate linear interpolation.

Solution: Consider the following values, use direct multivariate linear interpolation to calculate $z(x,y) = z(1100, 1225)$,

y	x		
	800	1000	1200
1150	1380.4	1500.2	1614.5
1200	1377.7	1499.0	1613.6
1250	1375.2	1497.1	1612.6

The form of the approximating polynomial is

$$z = f(x, y) = a + bx + cy + dxy$$

Substituting the four data points that bracket $x = 1100$ and $y = 1225$ into the polynomial gives

$$1499.0 = a + (1000)b + (1200)c + (1000)(1200)d$$

$$1497.1 = a + (1000)b + (1250)c + (1000)(1250)d$$

$$1613.6 = a + (1200)b + (1200)c + (1200)(1200)d$$

$$1612.6 = a + (1200)b + (1250)c + (1200)(1250)d$$

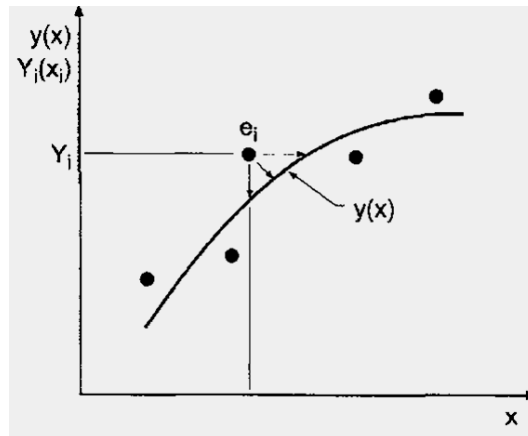
Solving for a , b , c , and d by Gauss elimination yields

$$z = f(x, y) = 1079.6 + 0.4650x - 0.1280y + 0.0900 \times 10^{-3} xy$$

Substitute $x = 1100$ and $y = 1225$ in the approximating polynomial gives $z(1100, 1225) = 1555.5$. The error in this result is $\text{Error} = 1555.5 - 1556.0 = -0.5$. The advantage of this approach is that we can use the same approximating polynomial to evaluate other values of $z(x,y)$, if required, without reevaluating the polynomial coefficients.

Least Squares Approximation

An approximate fit yields a polynomial that passes through the set of points in the best possible manner without being required to pass exactly through any of the points. Several definitions of best possible manner exist. Consider the set of discrete points, $[x_i, Y(x_i)] = (x_i, Y_i)$, and the approximate polynomial $y(x)$ chosen to represent the set of discrete points, as illustrated in the figure below. The discrete points do not fall on the approximating polynomial. The deviations (i.e., distances) of the points from the approximating function must be minimized in some manner.

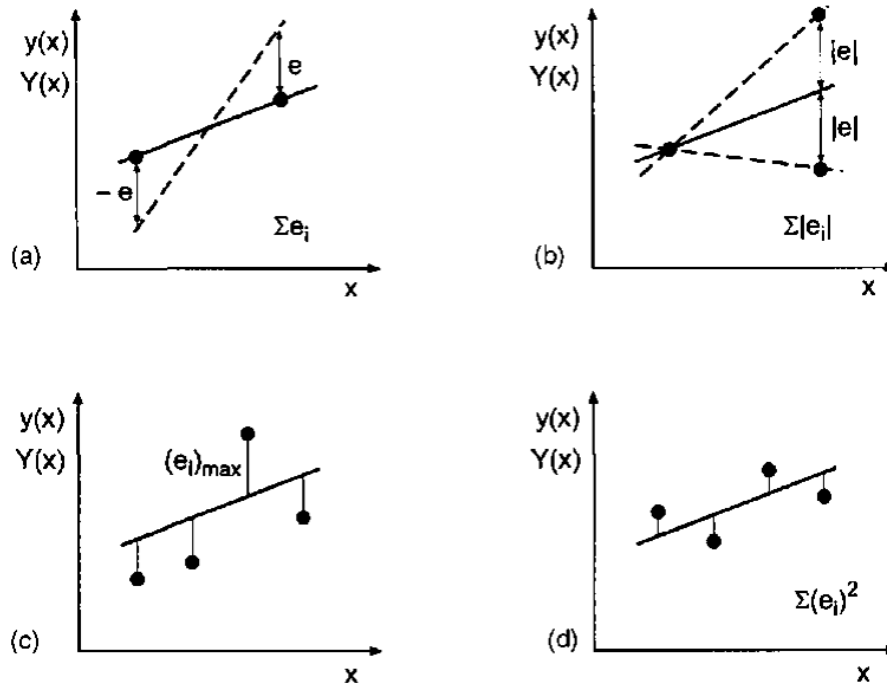


If the values of the independent variable x_i are considered exact, then all the deviation is assigned to the dependent variable Y_i , and the deviation e_i is the vertical distance between Y_i and $y_i = f(x_i)$. Thus,

$$e_i = Y_i - y_i$$

It is certainly possible that the values of Y_i are quite accurate, but the corresponding values of x_i are in error. In that case, the deviation would be measured by the horizontal distance illustrated in the above figure. If x_i and Y_i both have uncertainties in their values, then the perpendicular distance between a point and the approximating function would be the deviation. The usual approach in the approximate fitting of tabular data is to assume that the deviation is the vertical distance between a point and the approximating function, as specified by the above equation.

Several best criteria are illustrated in the next figure for a straight-line approximation. From the figure we can see that the least squares criteria, in which the sum of the squares of the deviations is minimized. The least squares procedure yields a good compromise criterion for the best fit approximation.



Best fit criteria. (a) Minimize $\sum e_i$. (b) Minimize $\sum |e_i|$. (c) Minimax. (d) Least squares.

The Straight-Line Approximation

The simplest polynomial is a linear polynomial, the straight line. Least squares straight line approximations are an extremely useful and common approximate fit, which is determined as follows. Given N data points, (x_i, Y_i) , fit the best straight line through the set data. The approximating function is

$$y = a + b x \quad \dots\dots\dots (1)$$

At each value of x_i , Eq. (1) gives

$$y_i = a + b x_i \quad (i = 1, \dots, N)$$

The deviation e_i at each value of x_i is

$$e_i = Y_i - y_i \quad (i = 1, \dots, N)$$

The sum of the squares of the deviations defines the function $S(a,b)$:

$$S(a,b) = \sum_{i=1}^N (e_i)^2 = \sum_{i=1}^N (Y_i - a - bx_i)^2$$

The function $S(a,b)$ is a minimum when $\partial S / \partial a = \partial S / \partial b = 0$. Thus,

$$\left. \begin{aligned} \frac{\partial S}{\partial a} &= \sum_{i=1}^N 2(Y_i - a - bx_i)(-1) = 0 \\ \frac{\partial S}{\partial b} &= \sum_{i=1}^N 2(Y_i - a - bx_i)(-x_i) = 0 \end{aligned} \right| \dots\dots\dots (2)$$

Dividing Eq. (2) by 2 and rearranging yields

$$\begin{aligned} aN + b \sum_{i=1}^N x_i &= \sum_{i=1}^N Y_i \\ a \sum_{i=1}^N x_i + b \sum_{i=1}^N x_i^2 &= \sum_{i=1}^N x_i Y_i \end{aligned}$$

Which is called the **normal equations** of the least squares fit. They can be solved for a and b by Gauss elimination.

Example: Least squares straight line approximation.

Solution: Consider the constant pressure specific heat for air at low temperatures presented in the following table, where T is the temperature and C_p is the specific heat. Determine a least squares straight line approximation for the set of data:

T	300	400	500	600	700	800	900	1000
C_p	1.0045	1.0134	1.0296	1.0507	1.0743	1.0984	1.1212	1.1410

$$C_p = a + bT$$

For this problem the equation becomes

$$8a + b \sum_{i=1}^8 T_i = \sum_{i=1}^8 C_{p,i}$$

$$a \sum_{i=1}^8 T_i + b \sum_{i=1}^8 T_i^2 = \sum_{i=1}^8 T_i C_{p,i}$$

Evaluating the summations and substituting into the above equations gives

$$8a + 5200b = 8.5331$$

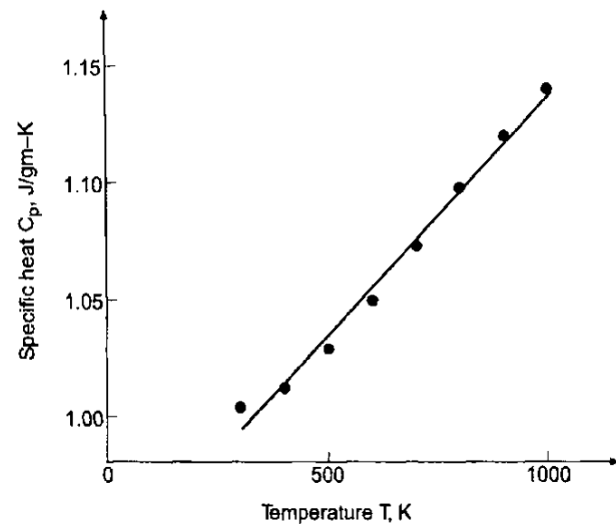
$$5200a + 3,800,000b = 5632.74$$

Solving for a and b by Gauss elimination without scaling or pivoting yields

$$C_p = 0.933194 + 0.205298 \times 10^{-3} T$$

Substituting the initial values of T into this equation gives the results presented in the next table and figure, which presents the exact data, the least squares straight line approximation, and the percent error. The straight line is not a very good approximation of the data.

T, K	$C_{p, \text{exact}}$	$C_{p, \text{approx}}$	Error, %
300	1.0045	0.9948	-0.97
400	1.0134	1.0153	0.19
500	1.0296	1.0358	0.61
600	1.0507	1.0564	0.54
700	1.0743	1.0769	0.24
800	1.0984	1.0974	-0.09
900	1.1212	1.1180	-0.29
1000	1.1410	1.1385	-0.22



High-Degree Polynomial Approximation

Given the N data points, (x_i, Y_i) , fit the best n th-degree polynomial through the set of data. Consider the n th-degree polynomial:

$$y = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

The sum of the squares of the deviation is given by

$$S(a_0, a_1, \dots, a_n) = \sum_{i=1}^N (e_i)^2 = \sum_{i=1}^N (Y_i - a_0 - a_1 x_i - \dots - a_n x_i^n)^2$$

The function $S(a_0, a_1, \dots, a_n)$ is a minimum when

$$\frac{\partial S}{\partial a_0} = \sum_{i=1}^N 2 (Y_i - a_0 - a_1 x_i - \dots - a_n x_i^n) (-1) = 0$$

.....

$$\frac{\partial S}{\partial a_n} = \sum_{i=1}^N 2 (Y_i - a_0 - a_1 x_i - \dots - a_n x_i^n) (-x_i^n) = 0$$

Dividing by 2 and rearranging yields the normal equations:

$$\begin{aligned} a_0 N + a_1 \sum_{i=1}^N x_i + \dots + a_n \sum_{i=1}^N x_i^n &= \sum_{i=1}^N Y_i \\ \dots & \\ a_0 \sum_{i=1}^N x_i^n + a_1 \sum_{i=1}^N x_i^{n+1} + \dots + a_n \sum_{i=1}^N x_i^{2n} &= \sum_{i=1}^N x_i^n Y_i \end{aligned}$$

And this can be solved for a_0 to a_n by Gauss elimination.

Example: Least squares quadratic polynomial approximation.

Solution: Determine a least squares quadratic polynomial approximation for this set of data:

x	1000	1500	2000	2500	3000
Y	1.1410	1.2095	1.2520	1.2782	1.2955

$$y = a + bx + cx^2$$

For this problem the polynomial becomes

$$aN + b \sum x_i + c \sum x_i^2 = \sum Y_i$$

$$a \sum x_i + b \sum x_i^2 + c \sum x_i^3 = \sum x_i Y_i$$

$$a \sum x_i^2 + b \sum x_i^3 + c \sum x_i^4 = \sum x_i^2 Y_i$$

Or may be represented in matrix form

$$\begin{bmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum Y_i \\ \sum x_i Y_i \\ \sum x_i^2 Y_i \end{bmatrix}$$

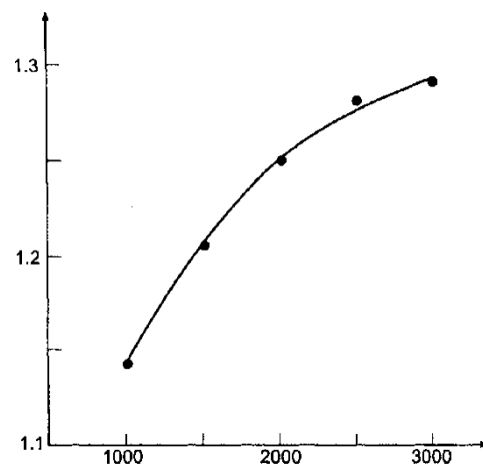
$$\begin{bmatrix} 5 & 10 \times 10^3 & 22.5 \times 10^6 \\ 10 \times 10^3 & 22.5 \times 10^6 & 55 \times 10^9 \\ 22.5 \times 10^6 & 55 \times 10^9 & 142.125 \times 10^{12} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 6.1762 \\ 12.5413 \times 10^3 \\ 288.5186 \times 10^6 \end{bmatrix}$$

Solving for **a**, **b**, and **c** by Gauss elimination yields

$$y = 0.965460 + 0.211197 \times 10^{-3} x - 0.0339143 \times 10^{-6} x^2$$

Substituting the initial values of **x** into the approximating polynomial gives the results presented in the next table and figure. The quadratic polynomial is a reasonable approximation of the discrete data.

T, K	Y_{exact}	y_{approx}	Error, %
1000	1.1410	1.1427	0.15
1500	1.2095	1.2059	-0.29
2000	1.2520	1.2522	0.02
2500	1.2782	1.2815	0.26
3000	1.2955	1.2938	-0.13



Multivariate Polynomial Approximation

Many problems arise in engineering and science where the dependent variable is a function of two or more independent variables, for example, $z = f(x, y)$ is a two-variable, or bivariate, function.

Given the N data points, (x_i, y_i, Z_i) , fit the best linear bivariate polynomial through the set of data. Consider the linear polynomial:

$$z = a + bx + cy$$

The sum of the squares of the deviations is given by

$$S(a, b, c) = \sum (e_i)^2 = \sum (Z_i - a - bx_i - cy_i)^2$$

The function $S(a, b, c)$ is a minimum when

$$\frac{\partial S}{\partial a} = \sum 2 (Z_i - a - bx_i - cy_i)(-1) = 0$$

$$\frac{\partial S}{\partial b} = \sum 2 (Z_i - a - bx_i - cy_i)(-x_i) = 0$$

$$\frac{\partial S}{\partial c} = \sum 2 (Z_i - a - bx_i - cy_i)(-y_i) = 0$$

Dividing by 2 and rearranging yields the normal equations:

$$aN + b \sum_{i=1}^N x_i + c \sum_{i=1}^N y_i = \sum_{i=1}^N Z_i$$

$$a \sum_{i=1}^N x_i + b \sum_{i=1}^N x_i^2 + c \sum_{i=1}^N x_i y_i = \sum_{i=1}^N x_i Z_i$$

$$a \sum_{i=1}^N y_i + b \sum_{i=1}^N x_i y_i + c \sum_{i=1}^N y_i^2 = \sum_{i=1}^N y_i Z_i$$

Which can be solved for a , b , and c by Gauss elimination.

A linear fit to a set of bivariate data may be inadequate. Consider the quadratic bivariate polynomial:

$$z = a + bx + cy + dx^2 + ey^2 + fxy$$

The sum of the squares of the deviations is given by

$$S(a,b,c,d,e,f) = \sum (e_i)^2 = \sum (Z_i - a - bx_i - cy_i - dx_i^2 - ey_i^2 - f x_i y_i)^2$$

The function $S(a,b,c,d,e,f)$ is a minimum when

$$\frac{\partial S}{\partial a} = \sum 2 (Z_i - a - bx_i - cy_i - dx_i^2 - ey_i^2 - f x_i y_i)(-1) = 0$$

.....

$$\frac{\partial S}{\partial f} = \sum 2 (Z_i - a - bx_i - cy_i - dx_i^2 - ey_i^2 - f x_i y_i)(-x_i y_i) = 0$$

Dividing by 2 and rearranging yields the normal equations:

$$a N + b \sum x_i + c \sum y_i + d \sum x_i^2 + e \sum y_i^2 + f \sum x_i y_i = \sum Z_i$$

$$a \sum x_i + b \sum x_i^2 + c \sum x_i y_i + d \sum x_i^3 + e \sum x_i y_i^2 + f \sum x_i^2 y_i = \sum x_i Z_i$$

$$a \sum y_i + b \sum x_i y_i + c \sum y_i^2 + d \sum x_i^2 y_i + e \sum y_i^3 + f \sum x_i y_i^2 = \sum y_i Z_i$$

$$a \sum x_i^2 + b \sum x_i^3 + c \sum x_i^2 y_i + d \sum x_i^4 + e \sum x_i^2 y_i^2 + f \sum x_i^3 y_i = \sum x_i^2 Z_i$$

$$a \sum y_i^2 + b \sum x_i y_i^2 + c \sum y_i^3 + d \sum x_i^2 y_i^2 + e \sum y_i^4 + f \sum x_i y_i^3 = \sum y_i^2 Z_i$$

$$a \sum x_i y_i + b \sum x_i^2 y_i + c \sum x_i y_i^2 + d \sum x_i^3 y_i + e \sum x_i y_i^3 + f \sum x_i^2 y_i^2 = \sum x_i y_i Z_i$$

Which can be solved for a to f by Gauss elimination.

Example: Least squares quadratic bivariate polynomial approximation.

Solution: Consider the following values, use Least squares quadratic bivariate polynomial to calculate $z(x,y) = z(1100, 1225)$,

y	x		
	800	1000	1200
1150	1380.4	1500.2	1614.5
1200	1377.7	1499.0	1613.6
1250	1375.2	1497.1	1612.6

The form of the approximating polynomial is

$$z = a + b x + c y + d x^2 + e y^2 + f xy$$

$$\begin{bmatrix} N & \sum x_i & \sum y_i & \sum x_i^2 & \sum y_i^2 & \sum x_i y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i & \sum x_i^3 & \sum x_i y_i^2 & \sum x_i^2 y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 & \sum x_i^2 y_i & \sum y_i^3 & \sum x_i y_i^2 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^2 y_i & \sum x_i^4 & \sum x_i^2 y_i^2 & \sum x_i^3 y_i \\ \sum y_i^2 & \sum x_i y_i^2 & \sum y_i^3 & \sum x_i^2 y_i^2 & \sum y_i^4 & \sum x_i y_i^3 \\ \sum x_i y_i & \sum x_i^2 y_i & \sum x_i y_i^2 & \sum x_i^3 y_i & \sum x_i y_i^3 & \sum x_i^2 y_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} \sum Z_i \\ \sum x_i Z_i \\ \sum y_i Z_i \\ \sum x_i^2 Z_i \\ \sum y_i^2 Z_i \\ \sum x_i y_i Z_i \end{bmatrix}$$

Evaluating the summation and substituting in the matrix yields

$$\begin{bmatrix} 9 E0 & 10.800 E3 & 9.000 E3 & 12.975 E6 & 9.240 E6 & 10.800 E6 \\ 10.800 E3 & 12.975 E6 & 10.800 E6 & 15.606 E9 & 11.088 E9 & 12.975 E9 \\ 9.000 E3 & 10.800 E6 & 9.240 E6 & 12.975 E9 & 9.720 E9 & 11.088 E9 \\ 12.975 E6 & 15.606 E9 & 12.975 E9 & 18.792 E12 & 13.321 E12 & 15.606 E12 \\ 9.240 E6 & 11.088 E9 & 9.720 E9 & 13.321 E12 & 10.450 E12 & 11.664 E12 \\ 10.800 E6 & 12.975 E9 & 11.088 E9 & 15.606 E12 & 11.664 E12 & 13.321 E12 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 13.4703 E3 \\ 16.1638 E6 \\ 13.6118 E6 \\ 19.4185 E9 \\ 14.1122 E9 \\ 16.3337 E9 \end{bmatrix}$$

A problem arises for high-degree polynomials. The coefficients in the matrix are varying over a range of several orders of magnitude, which gives rise to ill-conditioned system. Normalizing each equation helps the situation. Double precision calculations are required.

Each row in the matrix should be normalized by the exponential term in the first coefficient of each row. Solving the normalized equations by Gauss eliminations yields

$$z(x, y) = 914.033 + 0.645500x - 0.020500y - 0.0000775x^2 - 0.000040y^2 + 0.0000825xy$$

Evaluating $z(1100, 1225) = 1556.3$.

The error is, **Error** = $1556.3 - 1556.0 = 0.3$, which is smaller than the error incurred by interpolation using direct multivariate linear approximation.

Numerical Analysis

DWE3214

PART-II: Numerical Solutions of ODE

Unit-6: Initial Value Problem



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-6: Initial Value Problem

Introduction

A classic example of an initial-value **ODE** is the general nonlinear first order **ODE**:

$$y' = f(t, y) \quad y(t_0) = y_0$$

This equation applies to many problems in engineering and science. Consider the lumped mass **m** illustrated in the figure. Heat transfer from the lumped mass **m** to its surroundings by radiation is governed by the Stefan-Boltzmann law of radiation:

$$\dot{q}_r = A \varepsilon \sigma (T^4 - T_a^4)$$

where \dot{q}_r is the heat transfer rate (J/s), A is the surface area of the lumped mass (m^2), ε is the Stefan-Boltzmann constant ($5.67 \times 10^{-8} \text{ J/m}^2\text{-K}^4\text{-s}$), σ is the emissivity of the body (dimensionless), which is the ratio of the actual radiation to the radiation from a black body, T is the internal temperature of the lumped mass (K), and T_a is the ambient temperature (K) (i.e., temperature of the surroundings). The energy E stored in the lumped mass is given by

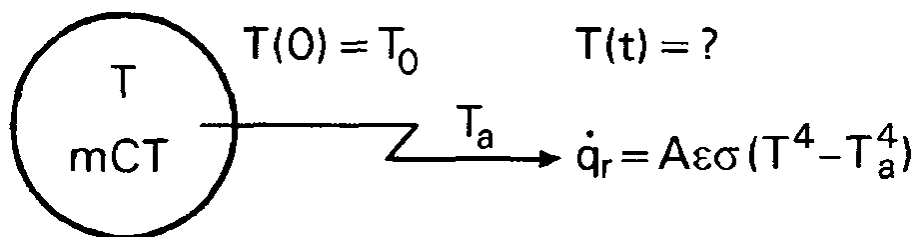
$$E = m C T$$

where m is the mass of the lumped mass (kg) and C is the specific heat of the material (J/kg-K). An energy balance states that the rate at which the energy stored in the lumped mass changes is equal to the rate at which heat is transferred to the surroundings. Thus,

$$\frac{d(mCT)}{dt} = -\dot{q}_r = -A \varepsilon \sigma (T^4 - T_a^4)$$

The minus sign is required so that the rate of change of stored energy is negative when T is greater than T_a . For constant m and C , the last equation can be written as

$$\frac{dT}{dt} = T' = -\alpha (T^4 - T_a^4) \quad \text{where} \quad \alpha = \frac{A \varepsilon \sigma}{mC}$$



Heat transfer by radiation from a lumped mass

Consider the case where the temperature of the surroundings is constant and the initial temperature of the lumped mass is $T(0.0)=T_0$. The initial-value problem is stated as follows:

$$T' = -\alpha(T^4 - T_a^4) = f(t, T) \quad T(0) = T_0$$

This is a nonlinear first-order initial-value **ODE**. The solution of this equation is the function $T(t)$, which describes the temperature history of the lumped mass corresponding to the initial conditions, $T(0.0)=T_0$.

An example of a higher-order initial-value **ODE** is given by the nonlinear second-order **ODE** governing the vertical flight of a rocket. The physical system is illustrated in the figure. Applying Newton's second law of motion, $\sum F = ma$, yields

$$\sum F = T - Mg - D = Ma = MV' = My''$$

where T is the thrust developed by the rocket motor (N), M is the instantaneous mass of the rocket (kg), g is the acceleration of gravity (m/s^2), which depends on the altitude y (m), D is the aerodynamic drag (N), a is the acceleration of the rocket (m/s^2), V is the velocity of the rocket (m/s), and y is the altitude of the rocket (m). The initial velocity, $V(0.0) = V_0$, is zero, and the initial elevation, $y(0.0) = y_0$, is zero. Thus, the initial conditions for the equation are

$$V(0.0) = y'(0.0) = 0.0 \quad \text{and} \quad y(0.0) = 0.0$$



$$\sum F = T - Mg - D = Ma = MV' = My''$$

$$y(0.0) = 0.0 \quad \text{and} \quad V(0.0) = 0.0$$

$$y(t) = ? \quad \text{and} \quad V(t) = ?$$

Vertical flight of a rocket

In general, the thrust T is a variable, which depends on time and altitude. The instantaneous mass M is given by

$$M(t) = M_o - \int_0^t \dot{m}(t) dt$$

where M_o is the initial mass of the rocket (kg), and $\dot{m}(t)$ is the instantaneous mass flow rate being expelled by the rocket (kg/s). The instantaneous aerodynamic drag D is given by

$$D(\rho, V, y) = C_D(\rho, V, y) \frac{1}{2} \rho(y) A V^2$$

where C_D is an empirical drag coefficient (dimensionless), which depends on the rocket geometry, the rocket velocity V and the properties of the atmosphere at altitude y (m); ρ is the density of the atmosphere (kg/m^3), which depends on the altitude y (m); and A is the cross-sectional frontal area of the rocket (m^2).

Combining the last equations yields the following second-order nonlinear initial-value ODE:

$$y'' = \frac{F(t, y)}{M_o - \int_0^t \dot{m}(t) dt} - g(y) - \frac{C_D(\rho, V, y) \frac{1}{2} \rho(y) A V^2}{M_o - \int_0^t \dot{m}(t) dt}$$

Consider a simpler model where T , \dot{m} , and g are constant, and the aerodynamic drag D is neglected. In that case, the last equation becomes

$$y'' = \frac{F}{M_o - \dot{m}t} - g \quad y(0.0) = 0.0 \quad \text{and} \quad y'(0.0) = V(0.0) = 0.0$$

The solution of the last two equations is the function $y(t)$, which describes the vertical motion of the rocket as a function of time t .

One-Dimensional Initial-Value Ordinary Differential Equations

The initial-value **ODEs** govern propagation problems, which are initial-value problems in open domains. Consequently, initial-value **ODEs** are solved numerically by marching methods. This section is devoted to presenting the basic properties of finite difference methods for solving initial-value (i.e., propagation) problems and to developing several specific finite difference methods.

The objective of a finite difference method for solving an ordinary differential equation (**ODE**) is to transform a calculus problem into an algebra problem by:

1. **Discretizing** the continuous physical domain into a discrete finite difference grid.
2. **Approximating** the exact derivatives in the **ODE** by algebraic finite difference approximations (**FDAs**).
3. **Substituting** the **FDAs** into the **ODE** to obtain an algebraic finite difference equation (**FDE**).
4. **Solving** the resulting algebraic **FDE**.

Finite difference approximations

Now that the finite difference grid has been specified, finite difference approximations (**FDAs**) of the exact derivatives in the **ODE** must be developed. This is accomplished using the Taylor series approach developed in Chapter 4.

In the development of finite difference approximations of differential equations, a distinction must be made between the exact solution of the differential equation and the solution of the finite difference equation which approximates the exact differential equation. For the remainder of this chapter, the exact solution of the **ODE** is denoted by an overbar on the symbol for the dependent variable [i.e., $\bar{y}(t)$], and the approximate solution is denoted by the symbol for the dependent variable without an overbar [i.e., $y(t)$]. Thus,

$\bar{y}(t)$ = exact solution

$y(t)$ = approximate solution

Exact derivatives, such as \bar{y}' , can be approximated at a grid point in terms of the values of \bar{y} at that grid point and adjacent grid points in several ways. Consider the derivative \bar{y}' .

Writing the Taylor series for \bar{y}_{n+1} using grid point n as the base point gives

$$\bar{y}_{n+1} = \bar{y}_n + \bar{y}'|_n \Delta t + \frac{1}{2} \bar{y}''|_n \Delta t^2 + \frac{1}{6} \bar{y}'''|_n \Delta t^3 + \dots$$

This equation can be expressed as the Taylor polynomial with remainder:

$$\bar{y}_{n+1} = \bar{y}_n + \bar{y}'|_n \Delta t + \frac{1}{2} \bar{y}''|_n \Delta t^2 + \dots + \frac{1}{m!} \bar{y}^{(m)}|_n \Delta t^m + R^{m+1}$$

where the remainder R^{m+1} is given by

$$R^{m+1} = \frac{1}{(m+1)!} \bar{y}^{(m+1)}(\tau) \Delta t^{m+1}$$

where $t \leq \tau \leq t + \Delta t$. The remainder term is simply the next term in Taylor series evaluated at $t = \tau$. If the infinite Taylor series is truncated after the m^{th} derivative term to obtain an approximation of \bar{y}^{n+1} , the remainder term R_{m+1} is the error associated with the truncated Taylor series. In most cases, our main concern is the order of the error, which is the rate at which the error goes to zero as $\Delta t \rightarrow 0$.

Solving the last equation for $\bar{y}'|_n$ yields

$$\bar{y}'|_n = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2} \bar{y}''|_n \Delta t - \frac{1}{6} \bar{y}'''|_n \Delta t^2 - \dots$$

If this equation is terminated after the first term on the right-hand side, it becomes

$$\bar{y}'|_n = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2} \bar{y}''(\tau) \Delta t$$

A finite difference approximation of $\bar{y}'|_n$, which will be denoted by $y'|_n$, can be obtained from the last equation by truncating the remainder term. Thus,

$y' _n = \frac{y_{n+1} - y_n}{\Delta t} \quad 0(\Delta t)$
--

where $0(\Delta t)$ term is shown to remind us of the order of the remainder term, which was truncated, which is the order of the approximation of $\bar{y}'|_n$. The remainder term which has been truncated to obtain the last equation is called the **truncation error** of the finite difference approximation of $\bar{y}'|_n$. This equation is a first order forward-difference approximation of \bar{y}' at grid point n .

A first order backward-difference approximation of \bar{y}' at grid point $n+1$ can be obtained by writing the Taylor series for \bar{y}_n using grid point $n+1$ as the base point and solving for $\bar{y}'|_{n+1}$. Thus,

$$\bar{y}_n = \bar{y}_{n+1} + \bar{y}'|_{n+1}(-\Delta t) + \frac{1}{2}\bar{y}''|_{n+1}(-\Delta t)^2 + \dots$$

$$\bar{y}'|_{n+1} = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} + \frac{1}{2}\bar{y}''(\tau)\Delta t$$

Truncating the remainder term yields

$$\bar{y}'|_{n+1} = \frac{y_{n+1} - y_n}{\Delta t} \quad O(\Delta t)$$

A second-order centered-difference approximation of \bar{y}' at grid point $n + \frac{1}{2}$ can be obtained by writing the Taylor series for \bar{y}_{n+1} and \bar{y}_n using grid point $n + \frac{1}{2}$ as the base point, subtracting the two Taylor series, and solving for $\bar{y}'|_{n+\frac{1}{2}}$. Thus,

$$\bar{y}_{n+1} = \bar{y}_{n+\frac{1}{2}} + \bar{y}'_{n+\frac{1}{2}}\left(\frac{\Delta t}{2}\right) + \frac{1}{2}\bar{y}''_{n+\frac{1}{2}}\left(\frac{\Delta t}{2}\right)^2 + \frac{1}{6}\bar{y}'''_{n+\frac{1}{2}}\left(\frac{\Delta t}{2}\right)^3 + \dots$$

$$\bar{y}_n = \bar{y}_{n+\frac{1}{2}} + \bar{y}'_{n+\frac{1}{2}}\left(-\frac{\Delta t}{2}\right) + \frac{1}{2}\bar{y}''_{n+\frac{1}{2}}\left(-\frac{\Delta t}{2}\right)^2 + \frac{1}{6}\bar{y}'''_{n+\frac{1}{2}}\left(-\frac{\Delta t}{2}\right)^3 + \dots$$

Subtracting the second equation from the first one, and solving for $\bar{y}'_{n+\frac{1}{2}}$ yields

$$\bar{y}'_{n+\frac{1}{2}} = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{24}\bar{y}'''(\tau)\Delta t^2$$

Truncating the remainder term yields

$$y'_{n+\frac{1}{2}} = \frac{y_{n+1} - y_n}{\Delta t} \quad O(\Delta t)^2$$

Note that the three (forward-backward-centered) equations of y' are identical algebraic expressions. They all yield the same numerical value. The differences in the three finite difference approximations are the value of the truncation errors.

All the above equations can be applied to steady space marching problems simply by changing t to x in all the equations.

Occasionally a finite difference of an exact derivative is presented without its development. In such cases, the truncation error and order can be determined by a consistency analysis using Taylor series. For example, consider the following finite difference approximation (**FDA**):

$$FDA = \frac{y_{n+1} - y_n}{\Delta t}$$

The Taylor series for the approximate solution $y(t)$ with base point n is

$$y_{n+1} = y_n + y'_n \Delta t + \frac{1}{2} y''_n \Delta t^2 + \dots$$

Substituting the Taylor series for y_{n+1} into the **FDA**, yields

$$FDA = \frac{y_n + y'_n \Delta t + \frac{1}{2} y''_n \Delta t^2 + \dots - y_n}{\Delta t} = y'_n + \frac{1}{2} y''_n \Delta t + \dots$$

As $\Delta t \rightarrow 0$, **FDA** $\rightarrow y'_n$, which shows that FDA is an approximation of the exact derivative \bar{y} at grid point n . The order of **FDA** is $O(\Delta t)$. The exact form of the truncation error relative to grid point n is determined. Choosing other base points for the Taylor series yields the truncation errors relative to those base points.

A finite difference approximation (**FDA**) of an exact derivative is **consistent** with the exact derivative if the **FDA** approaches the exact derivative as $\Delta t \rightarrow 0$, as illustrated in the last equation. Consistency is an important property of finite difference approximation of derivatives.

Finite difference equations

Finite difference solutions of differential equations are obtained by discretizing the continuous solution domain and replacing the exact derivatives in the differential equation by finite difference approximations to obtain a finite approximation of the differential equation. Such approximations are called **finite difference equations (FDEs)**.

Consider the general nonlinear initial-value **ODE**:

$$\bar{y}' = f(t, \bar{y}) \quad \bar{y}(0) = \bar{y}_0$$

Choose a finite difference approximation (**FDA**), y' , for \bar{y}' . For example:

$$y'_n = \frac{y_{n+1} - y_n}{\Delta t} \quad \text{or} \quad y'_{n+1} = \frac{y_{n+1} - y_n}{\Delta t}$$

Substitute the **FDA** for \bar{y}' into the exact ODE, $\bar{y}' = f(t, \bar{y})$, and solve for y_{n+1} :

$$y'_n = \frac{y_{n+1} - y_n}{\Delta t} = f(t_n, y_n) = f_n$$

$$y'_{n+1} = \frac{y_{n+1} - y_n}{\Delta t} = f(t_{n+1}, y_{n+1}) = f_{n+1}$$

Solving the first equation for y_{n+1} yields

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) = y_n + \Delta t f_n \quad \dots\dots\dots (1)$$

Solving the second equation for y_{n+1} yields

$$y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1}) = y_n + \Delta t f_{n+1} \quad \dots\dots\dots (2)$$

Equation (1) is an **explicit** finite difference equation, since f_n does not depend on y_{n+1} , and it can be solved explicitly for y_{n+1} . Equation (2) is an **implicit** finite difference equation, since f_{n+1} depends on y_{n+1} . If the **ODE** is linear, then f_{n+1} is linear in y_{n+1} , and it can be solved directly for y_{n+1} . If the **ODE** is nonlinear, then f_{n+1} is nonlinear in y_{n+1} , and additional effort is required to solve it for y_{n+1} .

Smoothness

Smoothness refers to the continuity of a function and its derivatives. The finite difference method of solving a differential equation employs Taylor series to develop finite difference approximations (**FDAs**) of the exact derivatives in the differential equation. If a problem has discontinuous derivatives of some order at some point in the solution domain, then **FDAs** based on the Taylor series may misbehave at that point.

For example, consider the vertical flight of a rocket illustrated in Figure (II.6). When the rocket engine is turned off, the thrust drops to zero instantly. This causes a discontinuity in the acceleration of the rocket, which causes a discontinuity in the second derivative of the altitude $y(t)$. The solution is not smooth in the neighborhood of the discontinuity in the second derivative of $y(t)$.

At a discontinuity, single point methods or extrapolation methods should be employed since the step size in the neighborhood of the discontinuity can be chosen so that the

discontinuity occurs at a grid point. Multipoint methods should not be employed in the neighborhood of a discontinuity in the function or its derivatives.

*Problems which do not have any discontinuities in the function, or its derivatives are called **smoothly varying problems**. Problems which have discontinuities in the function, or its derivatives are called **non-smoothly varying problems**.*

The First-Order Euler Methods

The **explicit Euler method** and the **implicit Euler method** are two first-order finite difference methods for solving initial-value **ODEs**. Although these methods are too inaccurate to be of much practical value, they are useful to illustrate many concepts relevant to the finite difference solution of initial-value **ODEs**.

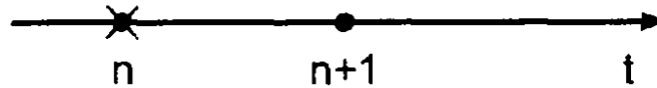
The Explicit Euler Method

Consider the general nonlinear first-order **ODE**:

$$\bar{y}' = f(t, \bar{y}) \quad \bar{y}(t_0) = \bar{y}_0$$

Choose point **n** as the base point and develop a finite difference approximation of this equation at that point. The finite difference grid is illustrated in the figure, where the cross (i.e., \times) denotes the base point for the finite difference approximation of the equation. The first order forward-difference finite difference approximation of \bar{y}' is given previously by

$$\bar{y}'|_n = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2} \bar{y}''(\tau) \Delta t$$



Finite difference grid for the explicit Euler method

Substituting this equation in the general nonlinear first-order **ODE** and evaluating $f(t, \bar{y})$ at point **n** yields

$$\frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2} \bar{y}''(\tau_n) \Delta t = f(t_n, \bar{y}_n) = \bar{f}_n$$

Solving for \bar{y}_{n+1} gives

$$\bar{y}_{n+1} = \bar{y}_n + \Delta t \bar{f}_n + \frac{1}{2} \bar{y}''(\tau_n) \Delta t^2 = \bar{y}_n + \Delta t \bar{f}_n + O(\Delta t^2)$$

Truncating the remainder term, which is $O(\Delta t^2)$, and solving for y_{n+1} yields the **explicit Euler finite difference equation (FDE)**:

$$y_{n+1} = y_n + \Delta t f_n + O(\Delta t^2)$$

where the $O(\Delta t^2)$ term is included as a remainder of the order of the local truncation error. Several features of this equation are summarized below:

1. The **FDE** is explicit, since f_n does not depend on y_{n+1} .
2. The **FDE** requires only one known point. Hence, it is a single point method.
3. The **FED** requires only one derivative function evaluation [i.e. $f(t, y)$] per step.
4. The error in calculating y_{n+1} for a single step, the local truncation error, is $O(\Delta t^2)$.
5. The global (i.e., total) error accumulated after N steps is $O(\Delta t)$. This result is derived in the following paragraph.

The **explicit Euler finite difference equation** is applied repetitively to march from the initial point t_0 to the final point, t_N , as illustrated in Figure (5.6). The solution at point N is

$$y_N = y_0 + \sum_{n=0}^{N-1} (y_{n+1} - y_n) = y_0 + \sum_{n=0}^{N-1} \Delta y_{n+1}$$

The total truncation error is given by

$$\text{Error} = y_0 + \sum_{n=0}^{N-1} \left(\frac{1}{2} y''(\tau_n) \Delta t^2 \right) = N \frac{1}{2} y''(\tau) \Delta t^2$$

where $t_0 \leq \tau \leq t_N$. The number of steps N is related to the step size Δt as follows:

$$N = \frac{t_N - t_0}{\Delta t}$$

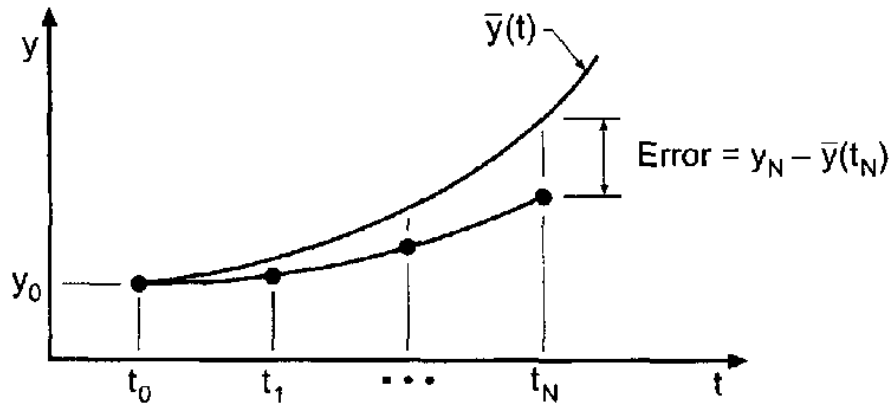
Substituting the last equation in the error equation yields

$$\text{Error} = \frac{1}{2} (t_N - t_0) y''(\tau) \Delta t = O(\Delta t)$$

Consequently, the global (i.e., total) error of the explicit Euler FDE is $O(\Delta t)$, which is the same as the order of the finite difference approximation of the exact derivative \bar{y}' , which is $O(\Delta t)$, as shown previously.

The result developed in the preceding paragraph applies to all finite difference approximations of first-order ordinary differential equations. The order of the global error is always equal to the order of the finite difference approximation of the exact derivative \bar{y}' .

The algorithm base on the repetitive application of the explicit Euler **FEE** to solve initial-value **ODEs** is called the **explicit Euler method**.



Repetitive application of the explicit Euler method.

Example: The explicit Euler method

Solution: Let's solve the radiation problem presented earlier using the **explicit Euler finite difference equation**. The derivative function is $f(t, T) = -\alpha(T^4 - T_a^4)$. Thus,

$$T_{n+1} = T_n - \Delta t \alpha (T_n^4 - T_a^4), \quad \alpha = -4.0 \times 10^{-12}, \quad T_a = 250, \quad T_0 = 2500$$

Let $\Delta t = 2.0$ s. For the first time step,

$$f_0 = -(4.0 \times 10^{-12})(2500.0^4 - 250.0^4) = -156.234375$$

$$T_1 = 2500.0 + 2.0(-156.234375) = 2187.531250$$

The results and the result of the subsequent time steps for t from 4.0s to 10.0s are summarized in the following table. The results for $\Delta t = 1.0$ s are also presented.

t_n t_{n+1}	T_n T_{n+1}	f_n	\tilde{T}_{n+1}	Error
0.0	2500.000000	-156.234375		
2.0	2187.531250	-91.580490	2248.247314	-60.716064
4.0	2004.370270	-64.545606	2074.611898	-70.241628
6.0	1875.279058	-49.452290	1944.618413	-69.339355
8.0	1776.374478	-39.813255	1842.094508	-65.720030
10.0	1696.747960		1758.263375	-61.515406
0.0	2500.000000	-156.234375		
1.0	2343.765625	-120.686999	2360.829988	-17.064363
2.0	2223.078626	-97.680938	2248.247314	-25.168688
3.0	2125.397688	-81.608926	2154.470796	-29.073108
.....
9.0	1768.780668	-39.136553	1798.227867	-29.447199
10.0	1729.644115		1758.263375	-28.619260

Several important features of the explicit Euler method are illustrated in that table. First, the solutions for both step sizes are following the general trend of the exact solution correctly. The solution for the smaller step size is more accurate than the solution for the larger step size. In fact, the order of the method can be estimated by comparing the errors at $t = 10.0s$. Thus,

$$E(\Delta t = 2.0) = \frac{1}{2}(t_N - t_0)T''(\tau)(2.0)$$

$$E(\Delta t = 1.0) = \frac{1}{2}(t_N - t_0)T''(\tau)(1.0)$$

Assuming that the value of $T''(\tau)$ are approximately equal, the ratio of the theoretical error is

$$\text{Ratio} = \frac{E(\Delta t = 2.0)}{E(\Delta t = 1.0)} = \frac{-61.515406}{-28.629260} = 2.15$$

The ratio shows that the method is first order. The value of 2.15 is not exactly equal to the theoretical value 2.0 due to the finite step size. The theoretical value of 2.0 is achieved only in the limit as $\Delta t \rightarrow 0$.

Another feature illustrated in the table is that the errors are relatively large. This is due to the large first-order, $O(\Delta t)$, truncation error. The errors are all negative, indicating that the numerical solution leads to the exact solution. This occurs because the derivative function $f(t, T)$ decreases as t increases as illustrated in the table. The derivative function in the **FDE** is evaluated at point n , the beginning of the interval of integration, where it has its largest value for the interval. Consequently, the numerical solution leads the exact solution.

The final feature of the explicit Euler method which is illustrated in the table is that the numerical solution approaches the exact solution as the step size decreases. This property of a finite difference method is called **convergence**. Convergence is necessary for a finite difference method to be of any use in solving a differential equation.

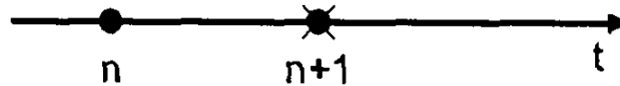
When the base point for the finite difference approximation of an **ODE** is point **n**, the unknown value y_{n+1} appears in the finite difference approximation of \bar{y}' , but not in the derivative function $f(t, \bar{y})$. Such **FDEs** are called **explicit FDEs**. The explicit Euler method is the simplest example of an explicit **FDE**.

When the base point for the finite difference approximation of an **ODE** is point **n+1**, the unknown value y_{n+1} appears in the finite difference approximation of \bar{y}' and in the derivative function $f(t, \bar{y})$. Such **FDEs** are called **implicit FDEs**.

The Implicit Euler Method

Consider the general nonlinear first-order **ODE**:

$$\bar{y}' = f(t, \bar{y}) \quad \bar{y}(t_0) = \bar{y}_0$$



Finite difference grid for the implicit Euler method.

Choose point **n+1** as the base point and develop a finite difference approximation of the above general nonlinear first-order **ODE** at that point. The finite difference grid is illustrated in Figure (5.7). The first-order backward difference finite difference approximation of \bar{y}' is given previously by:

$$\bar{y}'|_{n+1} = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} + \frac{1}{2} \bar{y}''(\tau_{n+1}) \Delta t$$

Substituting this equation into the general nonlinear first-order **ODE**, and evaluating $f(t, \bar{y})$ at point **n+1** yields

$$\frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} + \frac{1}{2} \bar{y}''(\tau_{n+1}) \Delta t = f(t_{n+1}, \bar{y}_{n+1}) = \bar{f}_{n+1}$$

Solving for \bar{y}_{n+1} gives

$$\bar{y}_{n+1} = \bar{y}_n + \Delta t \bar{f}_{n+1} - \frac{1}{2} \bar{y}''(\tau_{n+1}) \Delta t^2 = \bar{y}_n + \Delta t \bar{f}_{n+1} + O(\Delta t^2)$$

Truncating the $O(\Delta t^2)$ remainder term yields the **implicit Euler FDE**:

$y_{n+1} = y_n + \Delta t f_{n+1} \quad O(\Delta t^2)$
--

Several features of this equation are summarized below:

1. The **FDE** is implicit, since f_{n+1} depends on y_{n+1} . If $f(t, y)$ is linear in y , then f_{n+1} is linear in y_{n+1} , and this equation is a linear **FDE** which can be solved directly for y_{n+1} . If $f(t, y)$ is nonlinear in y , then the equation is a nonlinear **FDE**, and additional effort is required to solve for y_{n+1} .
2. The **FDE** is a single-point **FDE**.
3. The **FDE** requires only one derivative function evaluation per step if $f(t, y)$ is linear in y . If $f(t, y)$ is nonlinear in y , the equation is nonlinear in y_{n+1} , and several evaluations of the derivative function may be required to solve the nonlinear **FDE**.
4. The single-step truncation error is $O(\Delta t^2)$, and the global error is $O(\Delta t)$.

The algorithm based on repetitive application of the implicit Euler FDE to solve initial-value ODEs is called the **implicit Euler method**.

The derivative function $f(t, \bar{y})$ may be linear or nonlinear in \bar{y} . When $f(t, \bar{y})$ is linear in \bar{y} , the corresponding **FDE** is linear in y_{n+1} , for both explicit **FDEs** and implicit **FDEs**. When $f(t, \bar{y})$ is nonlinear in \bar{y} , explicit **FDEs** are still linear in y_{n+1} . However, implicit **FDEs** are nonlinear in y_{n+1} , and special procedures are required to solve for y_{n+1} . One of the procedures (**Newton's method**) is discussed in the next example.

Example: *The implicit Euler method.*

Solutions: *Let's solve the radiation problem presented earlier using the **implicit Euler finite difference equation**. The derivative function is $f(t, T) = -\alpha(T^4 - T_a^4)$. Thus,*

$$T_{n+1} = T_n + \Delta t f_{n+1}$$

$$T_{n+1} = T_n - \alpha \Delta t (T_{n+1}^4 - T_a^4)$$

*This equation is a nonlinear fourth-order polynomial **FDE**. Procedure for solving nonlinear implicit FDEs is presented in the following using **Newton's method**.*

Rearranging the last equation into the form of $y_{n+1} = G(y_{n+1})$ or,

$$F(y_{n+1}) = y_{n+1} - G(y_{n+1}) = 0$$

Expanding $F(y_{n+1})$ in a Taylor series about the value y_{n+1} and evaluating at \tilde{y}_{n+1} yields

$$F(\tilde{y}_{n+1}) = F(y_{n+1}) + F'(y_{n+1})(\tilde{y}_{n+1} - y_{n+1}) + \dots = 0$$

where \tilde{y}_{n+1} is the solution of $y_{n+1} = G(y_{n+1})$. Truncating the last equation after the first-order term and solving for y_{n+1} yields

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - \frac{F(y_{n+1}^{(k)})}{F'(y_{n+1}^{(k)})}$$

The last equation must be solved iteratively. Newton's method works well for nonlinear implicit FDEs. A good initial guess may be required.

Returning back to our example where:

$$T_{n+1} = T_n + \Delta t f_{n+1}$$

$$T_{n+1} = T_n - \alpha \Delta t (T_{n+1}^4 - T_a^4)$$

Rearranging the equation to be solved using Newton's method yields

$$F(T_{n+1}) = T_{n+1} - T_n + \Delta t \alpha (T_{n+1}^4 - T_a^4) = 0$$

The derivative of $F(T_{n+1})$ is

$$F'(T_{n+1}) = 1 + 4 \Delta t \alpha T_{n+1}^3$$

Then

$$T_{n+1}^{(k+1)} = T_{n+1}^{(k)} - \frac{F(T_{n+1}^{(k)})}{F'(T_{n+1}^{(k)})}$$

Let $\Delta t = 2.0$ s . For the first time step,

$$F(T_1) = T_1 - 2500.0 + (2.0)(4.0 \times 10^{-12})(2500.0^4 - 250.0^4)$$

$$F'(T_1) = 1 + 4(2.0)(4.0 \times 10^{-12})T_1^3$$

Let $T_1^{(0)} = 2500.0$ K . Then

$$\begin{aligned} F(T_1^{(0)}) &= 2500.0 - 2500.0 + (2.0)(4.0 \times 10^{-12})(2500.0^4 - 250.0^4) \\ &= 312.468250 \end{aligned}$$

$$F'(T_1^{(0)}) = 1 + 4(2.0)(4.0 \times 10^{-12})2500.0^3 = 1.500000$$

$$T_1^{(1)} = 2500.0 - \frac{312.468250}{1.500000} = 2291.687500$$

Repeating the procedure three times yields the converged result $T_1^{(4)} = 2282.785819$. These results are presented in the following table, along with the final results for the subsequent time steps from $t = 4.0$ s to 10.0 s. The results for $\Delta t = 1.0$ s are also presented.

Solution by Newton's Method

t_n t_{n+1} t_{n+1}	k	T_n T_{n+1} T_{n+1}	F_n	F'_n	\bar{T}_{n+1}	Error
0.0	0	2500.000000				
	1	2500.000000	312.468750	1.500000		
	2	2291.687500	12.310131	1.385138		
	3	2282.800203	0.019859	1.380674		
	4	2282.785819	0.000000	1.380667		
2.0		2282.785819			2248.247314	34.538505
4.0		2120.934807			2074.611898	46.322909
6.0		1994.394933			1944.618413	49.776520
8.0		1891.929506			1842.094508	49.834998
10.0		1806.718992			1758.263375	48.455617

Solution by the Implicit Euler Method

t_n t_{n+1}	T_n T_{n+1}	\bar{T}_{n+1}	Error
0.0	2500.000000		
2.0	2282.785819	2248.247314	34.538505
4.0	2120.934807	2074.611898	46.322909
6.0	1994.394933	1944.618413	49.776520
8.0	1891.929506	1842.094508	49.834998
10.0	1806.718992	1758.263375	48.455617
0.0	2500.000000		
1.0	2373.145960	2360.829988	12.315972
2.0	2267.431887	2248.247314	19.184573
.....
9.0	1824.209295	1798.227867	25.981428
10.0	1783.732059	1758.263375	25.468684

The results presented in the last table behave generally the same as the results presented in the explicit Euler method. An error analysis at $t = 10.0$ s gives

$$\text{Ratio} = \frac{E(\Delta t = 2.0)}{E(\Delta t = 1.0)} = \frac{48.455617}{25.468684} = 1.90$$

which shows that the method is first order. The errors are all positive, indicating that the numerical solution lags the exact solution. This result is in direct contrast to the error behavior of the explicit Euler method, where a leading error was observed. In the present case, the derivative function of the **FDE** is evaluated at point $n+1$, the end of the interval of

integration, where it has its smallest value. Consequently, the numerical solution lags the exact solution.

Comparisons of the Explicit and Implicit Euler Methods

The explicit Euler method and the implicit Euler method are both first-order [i.e., $O(\Delta t)$] methods. As illustrated in the last two examples, the errors in these two methods are comparable (although of opposite sign) for the same step size. For nonlinear **ODEs**, the explicit Euler method is straightforward, but the implicit Euler method yields a nonlinear **FDE**, which is more difficult to solve. So, what is the advantage, if any, of the implicit Euler method?

The implicit Euler method is **unconditionally stable**, whereas the explicit Euler method is **conditionally stable**. This difference can be illustrated by solving the linear first order homogeneous **ODE**

$$\bar{y}' + \bar{y} = 0 \quad \bar{y}(0) = 1$$

For which $\bar{f}(t, \bar{y}) = -\bar{y}$, by both methods. The exact solution is

$$\bar{y}(t) = e^{-t}$$

Solving the **ODE** by the **explicit Euler** method yields the following **FDE**:

$$y_{n+1} = y_n + \Delta t f_n = y_n + \Delta t (-y_n)$$

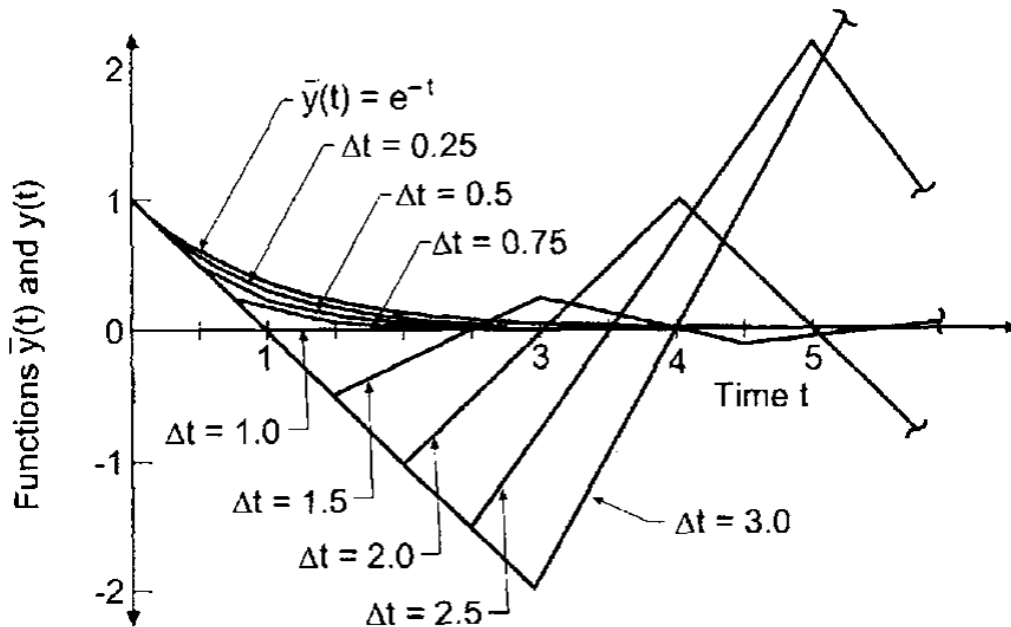
$$y_{n+1} = (1 - \Delta t) y_n$$

Solutions of this equation for several values of Δt are presented in Figure (5.8). The numerical solution behaves in a physically correct manner (i.e., decrease monolithically) for $\Delta t \leq 1.0$ as $t \rightarrow \infty$, and approaches the exact asymptotic solution, $\bar{y}(\infty) = 0$. For $\Delta t = 1.0$, the numerical solution reaches the exact asymptotic solution, $\bar{y}(\infty) = 0$, in one step.

For $1.0 < \Delta t < 2.0$, the numerical solution overshoots and oscillates about the exact asymptotic solution, $\bar{y}(\infty) = 0$, in a damped manner and approaches the exact asymptotic solution as $t \rightarrow \infty$. For $\Delta t = 2.0$, the numerical solution oscillates about the exact asymptotic solution in a stable manner but never approaches the exact asymptotic solution. Thus, solutions are stable for $\Delta t \leq 2.0$.

For $\Delta t > 2.0$, the numerical solution oscillates about the exact asymptotic solution in an unstable manner that grows exponentially without bound. This is **numerical instability**. Consequently, the explicit Euler method is **conditionally stable** for this **ODE**, that is, it is stable only for $\Delta t \leq 2.0$.

The oscillatory behavior for $1.0 < \Delta t < 2.0$ is called **overshoot** and must be avoided. Overshoot is not instability. However, it does not model physical reality, thus it is unacceptable. The step size Δt generally must be 50 percent or less of the stable step size to avoid overshoot.



Behavior of the explicit Euler method

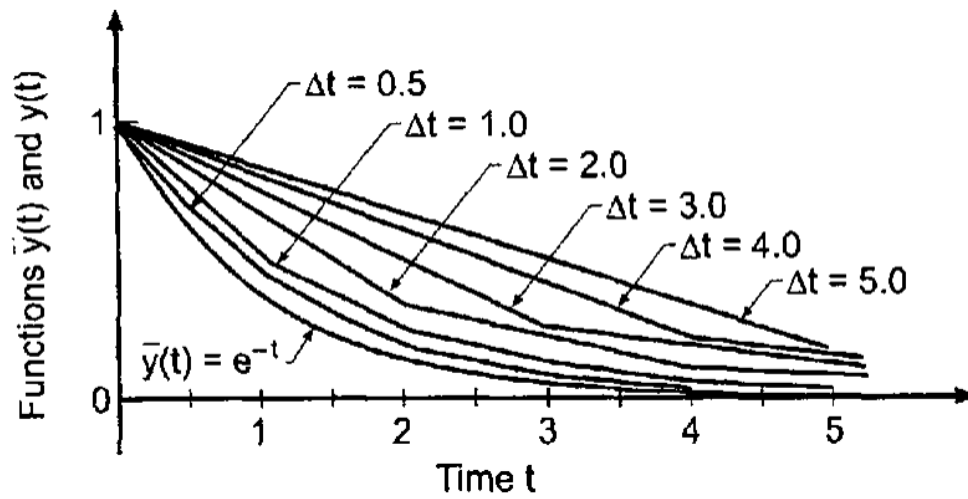
Solving the ODE by the **implicit Euler** method gives the following **FDE**:

$$y_{n+1} = y_n + \Delta t f_{n+1} = y_n + \Delta t (-y_{n+1})$$

This equation is linear in y_{n+1} , it can be solve directly for y_{n+1} to yield

$$y_{n+1} = \frac{y_n}{1 + \Delta t}$$

Which can be solved for several values of Δt as presented in the figure. The numerical solution behaves in a physically correct manner (i.e., decrease monotonically) for all values of Δt . This is **unconditional stability**, which is the main advantage of implicit methods. The error increase as Δt increases, but this is an accuracy problem, not a stability problem.



Behavior of the implicit Euler method

Runge-Kutta Methods

Runge-Kutta methods are a family of single-point methods which evaluate $\Delta y = y_{n+1} - y_n$ as the weighted sum of several Δy_i ($i = 1, 2, \dots$), where each Δy_i is evaluated as Δt multiplied by the derivative function $f(t, y)$, evaluated at the same point in the range $t_n \leq t \leq t_{n+1}$, and the C_i ($i = 1, 2, \dots$) are the weighting factors. Thus,

$$y_{n+1} = y_n + \Delta y_n = y_n + (y_{n+1} - y_n) \dots\dots\dots (23)$$

where Δy_n is given by

$\Delta y = C_1 \Delta y_1 + C_2 \Delta y_2 + C_3 \Delta y_3 + \dots\dots\dots$

(24)

The second order Runge-Kutta method is obtained by assuming that $\Delta y = y_{n+1} - y_n$ is a weighted sum of two Δy 's :

$y_{n+1} = y_n + C_1 \Delta y_1 + C_2 \Delta y_2$

(25)

where Δy_1 is given by the explicit Euler **FDE**:

$$\Delta y_1 = \Delta t f(t_n, y_n) = \Delta t f_n \dots\dots\dots (26)$$

and Δy_2 is based on $f(t, y)$ evaluated somewhere in the interval $t_n \leq t \leq t_{n+1}$:

$$\Delta y_2 = \Delta t f[t_n + (\alpha \Delta t), y_n + (\beta \Delta y_1)] \dots\dots\dots (27)$$

where α and β are to be determined. Let $\Delta t = h$. Substituting Δy_1 and Δy_2 into Eq. (25) gives

$$y_{n+1} = y_n + C_1 (h f_n) + C_2 h f[t_n + (\alpha \Delta t), y_n + (\beta \Delta y_1)] \dots\dots\dots (28)$$

Expressing $f(t, \bar{y})$ in a Taylor series at grid point **n** gives

$$f(t, \bar{y}) = \bar{f}_n + \bar{f}_t \Big|_n h + \bar{f}_y \Big|_n \Delta y + \dots\dots\dots (29)$$

Evaluating $f(t, \bar{y})$ at $t = t_n + (\alpha h)$ (i.e., $\Delta t = \alpha h$) and $y = y_n + (\beta \Delta y_n)$ (i.e., $\Delta y = \beta h f_n$) gives

$$f[t_n + (\alpha h), y_n + (\beta \Delta y_n)] = f_n + (\alpha h) f_t|_n + (\beta h f_n) f_y|_n + O(h^2) \quad \dots\dots\dots (30)$$

Substituting this result into Eq. (28) and collecting terms yields

$$y_{n+1} = y_n + (C_1 + C_2) h f_n + h^2 (\alpha C_2 f_t|_n + \beta C_2 f_n f_y|_n) + O(h^3) \quad \dots\dots\dots (31)$$

The four free parameters, C_1, C_2, α , and β can be determined by requiring Eq. (31) to match the Taylor series for $\bar{y}(t)$ through second-order terms. That series is

$$\bar{y}_{n+1} = \bar{y}_n + \bar{y}'|_n h + \frac{1}{2} \bar{y}''|_n h^2 + \dots\dots\dots (32)$$

$$\bar{y}'|_n = \bar{f}(t_n, \bar{y}_n) = \bar{f}_n \quad \dots\dots\dots (33)$$

$$\bar{y}''|_n = (\bar{y}')'|_n = \bar{f}'|_n = \frac{d\bar{f}}{dt}|_n = \bar{f}_t|_n + \bar{f}_y|_n \bar{y}'|_n + \dots\dots\dots (34)$$

Substituting Eqs. (34) and (33) into Eq. (32), where $\bar{y}'|_n = \bar{f}_n$, gives

$$\bar{y}_{n+1} = \bar{y}_n + h \bar{f}_n + \frac{1}{2} h^2 (\bar{f}_t|_n + \bar{f}_n \bar{f}_y|_n) + O(h^3) \quad \dots\dots\dots (35)$$

Equating Eqs. (31) and (35) term by term gives

$C_1 + C_2 = 1 \quad \alpha C_1 = \frac{1}{2} \quad \beta C_2 = \frac{1}{2}$	$\dots\dots\dots (36)$
--	------------------------

There are an infinite number of possibilities. Letting $C_1 = \frac{1}{2}$ gives $C_2 = \frac{1}{2}$, $\alpha = 1$, and $\beta = 1$, which yields the modified Euler **FDEs**. Thus,

$$\Delta y_1 = h f(t_n, y_n) = h f_n \quad \dots\dots\dots (37)$$

$$\Delta y_2 = h f(t_{n+1}, y_{n+1}) = h f_{n+1} \quad \dots\dots\dots (38)$$

$$y_{n+1} = y_n + \frac{1}{2} \Delta y_1 + \frac{1}{2} \Delta y_2 = y_n + \frac{h}{2} (f_n + f_{n+1}) \quad \dots\dots\dots (39)$$

Letting $C_1 = 0$ gives $C_2 = 1$, $\alpha = \frac{1}{2}$, and $\beta = \frac{1}{2}$, which yields the modified midpoint **FDEs**. Thus,

$$\Delta y_1 = h f(t_n, y_n) = h f_n \quad \dots\dots\dots (40)$$

$$\Delta y_2 = h f\left(t_n + \frac{h}{2}, y_n + \frac{\Delta y_1}{2}\right) = h f_{n+1/2} \quad \dots\dots\dots (41)$$

$$y_{n+1} = y_n + (0)\Delta y_1 + (1)\Delta y_2 = y_n + h f_{n+1/2} \quad \dots\dots\dots (42)$$

Other methods result for other choices for C_1 and C_2 .

In the general literature, Runge-Kutta formulas frequently denote the Δy_i 's by k 's ($i=1,2,\dots$). Thus, the second order Runge-Kutta **FDEs** which are identical to the modified Euler **FDEs**, Eqs. (37) to (39), are given by

$$y_{n+1} = y_n + \frac{1}{2} (k_1 + k_2) \quad \dots\dots\dots (43)$$

$$k_1 = h f(t_n, y_n) = h f_n \quad \dots\dots\dots (44)$$

$$k_2 = h f(t_n + \Delta t, y_n + k_1) = h f_{n+1} \quad \dots\dots\dots (45)$$

The Fourth Order Runge-Kutta Method

Runge-Kutta methods of higher order have been devised. One of the most popular is the following fourth-order method:

$$y_{n+1} = y_n + \frac{1}{6}(\Delta y_1 + 2\Delta y_2 + 2\Delta y_3 + \Delta y_4) \quad \dots\dots\dots (46)$$

$$\Delta y_1 = h f(t_n, y_n) \quad \Delta y_2 = h f\left(t_n + \frac{h}{2}, y_n + \frac{\Delta y_1}{2}\right) \quad \dots\dots (47.a)$$

$$\Delta y_3 = h f\left(t_n + \frac{h}{2}, y_n + \frac{\Delta y_2}{2}\right) \quad \Delta y_4 = h f(t_n + h, y_n + \Delta y_3) \quad \dots\dots (47.b)$$

To perform a consistency and order analysis and a stability analysis of the fourth order Runge-Kutta method, Eqs. (46) and (47) must be applied to the model **ODE**, $\bar{y}' + \alpha \bar{y} = 0$, for which $\bar{f}(t, \bar{y}) = -\alpha \bar{y}$, and the results must be combined into a single-step **FDE**. Thus,

$$\Delta y_1 = h f(t_n, y_n) = h(-\alpha y_n) = (-\alpha h)y_n \quad \dots\dots\dots (48)$$

$$\Delta y_2 = h f\left(t_n + \frac{h}{2}, y_n + \frac{\Delta y_1}{2}\right) = h\left(-\alpha \left\{y_n + \frac{1}{2}[-(\alpha h)y_n]\right\}\right) \quad \dots\dots\dots (49.a)$$

$$\Delta y_2 = -(\alpha h)y_n \left(1 - \frac{(\alpha h)}{2}\right) \quad \dots\dots\dots (49.b)$$

$$\Delta y_3 = h f\left(t_n + \frac{h}{2}, y_n + \frac{\Delta y_2}{2}\right) = h\left(-\alpha \left\{y_n + \frac{1}{2}\left[-(\alpha h)y_n \left(1 - \frac{(\alpha h)}{2}\right)\right]\right\}\right) \dots\dots (50.a)$$

$$\Delta y_3 = -(\alpha h)y_n \left[1 - \frac{(\alpha h)}{2} + \frac{(\alpha h)^2}{4}\right] \quad \dots\dots\dots (50.b)$$

$$\begin{aligned}\Delta y_4 &= h f(t_n + h, y_n + \Delta y_3) \\ &= h \left(-\alpha \left\{ y_n + \left[-(\alpha h) y_n \left(1 - \frac{(\alpha h)}{2} + \frac{(\alpha h)^2}{4} \right) \right] \right\} \right) \dots\dots\dots (51.a)\end{aligned}$$

$$\Delta y_4 = -(\alpha h) y_n \left[1 - (\alpha h) + \frac{(\alpha h)^2}{2} - \frac{(\alpha h)^3}{4} \right] \dots\dots\dots (51.b)$$

Substituting Eqs. (48), (49.b), (50.b), and (51.b) into Eq. (46) yields the single-step FDE corresponding to Eqs. (46) and (47):

$$y_{n+1} = y_n - (\alpha h) y_n + \frac{1}{2} (\alpha h)^2 y_n - \frac{1}{6} (\alpha h)^3 y_n + \frac{1}{24} (\alpha h)^4 y_n \dots\dots\dots (52)$$

In summary, the fourth order Runge-Kutta **FDEs** have the following characteristics:

1. The **FDEs** are an explicit predictor-corrector set of **FDEs** which requires two derivatives function evaluations per step.
2. The **FDEs** are consistent, $O(\Delta t^5)$ locally and $O(\Delta t^4)$ globally.
3. The **FDEs** are conditionally stable (i.e., $\alpha \Delta t \leq 2.875$).
4. The **FDEs** are consistent and conditionally stable, and thus, convergent.

algorithms based on the repetitive application of Runge-Kutta **FDEs** are called **Runge-Kutta methods**.

Example: The fourth order Runge-Kutta method

To illustrate the forth-order Runge-Kutta method, let's solve the radiation problem using Eq. (46) and (47). The derivative function is $f(t, T) = -\alpha(T^4 - T_a^4)$. Equations (46) and (47) yield

$$T_{n+1} = T_n + \frac{1}{6} (\Delta T_1 + 2\Delta T_2 + 2\Delta T_3 + \Delta T_4)$$

$$\Delta T_1 = \Delta t f(t_n, T_n) = \Delta t f_n \quad \Delta T_2 = \Delta t f\left(t_n + \frac{\Delta t}{2}, T_n + \frac{\Delta T_1}{2}\right)$$

$$\Delta T_3 = \Delta t f\left(t_n + \frac{\Delta t}{2}, T_n + \frac{\Delta T_2}{2}\right) \quad \Delta T_4 = \Delta t f(t_n + \Delta t, T_n + \Delta T_3)$$

Let $\Delta t = 2.0s$. For the first time step,

$$\Delta T_1 = (2.0)(-4.0 \times 10^{-12})(2500.0^4 - 250.0^4) = -312.46875000$$

$$\Delta T_2 = (2.0)(-4.0 \times 10^{-12}) \left[\left(2500.0 - \frac{312.46875000}{2} \right)^4 - 250.0^4 \right]$$

$$= -241.37399871$$

$$\Delta T_3 = (2.0)(-4.0 \times 10^{-12}) \left[\left(2500.0 - \frac{241.37399871}{2} \right)^4 - 250.0^4 \right]$$

$$= -256.35592518$$

$$\Delta T_4 = (2.0)(-4.0 \times 10^{-12}) \left[(2500.0 - 256.35592518)^4 - 250.0^4 \right]$$

$$= -202.69306346$$

$$T_1 = 2500.0 + \frac{1}{6} \left[-312.46875000 + 2(-241.37399871) \right. \\ \left. + 2(-256.35592518) - 202.69306346 \right] = 2248.22972313$$

These results, the results for subsequent time steps for $t = 4.0s$ to $10.0s$, and the solution for $\Delta t = 1.0s$ are presented in the next table.

Solution by the Fourth Order Runge-Kutta Method

t_n	T_n	ΔT_1	ΔT_2	Error
t_{n+1}	T_{n+1}	ΔT_3	ΔT_4	
0.0	2500.000000000	-312.468750000	-241.373998706	
		-256.355925175	-202.693063461	
2.0	2248.229723129	-204.335495214	-169.656671860	-0.017590925
		-175.210831240	-147.710427814	
4.0	2074.596234925	-148.160603003	-128.100880073	-0.015662958
		-130.689901700	-114.201682488	
6.0	1944.605593419	-114.366138259	-101.492235905	-0.012819719
		-102.884298188	-92.010660768	
8.0	1842.083948884	-92.083178136	-83.213391686	-0.010558967
		-84.038652301	-76.389312398	
10.0	1758.254519132			-0.008855569
0.0	2500.000000000	-156.234375000	-137.601503560	
		-139.731281344	-124.122675002	
1.0	2360.829563365	-124.240707542	-111.669705704	-0.000425090
		-112.896277161	-102.123860057	
2.0	2248.246807810			-0.000506244
...
9.0	1798.227583359	-41.809631378	-39.898370905	-0.000283433
		-39.984283934	-38.211873099	
10.0	1758.263114333			-0.000260369

The error at $t = 10.0s$ for $\Delta t = 1.0s$ is approximately 110,000 times smaller than the error presented for the first-order explicit Euler method and 3,500 smaller than the error presented for the solution by the modified Euler method. Results such as these clearly demonstrates the advantages of higher-order methods. An error analysis at $t = 10.0s$ gives

$$\text{Ratio} = \frac{E(\Delta t = 2.0)}{E(\Delta t = 1.0)} = \frac{-0.008855569}{-0.000260369} = 34.01$$

which demonstrates that the method is fourth order since the theoretical error ratio for an $O(\Delta t^4)$ method is 16.0.

Higher-order ordinary differential equations

Many applications in engineering and science are governed by higher-order ODEs. In general, a higher-order ODE can be replaced by a system of first-order ODEs. When a system of higher-order ODEs is involved, each individual higher-order ODE can be replaced by a system of first-order ODEs, and the coupled system of higher-order ODEs can be replaced by coupled system of first-order ODEs. The systems of the first-order ODEs can be solved as will be described later.

Consider the second-order initial value ODE developed for the vertical flight of a rocket, and simpler model given earlier

$$y'' = \frac{F(t, y)}{M_o - \int_0^t \dot{m}(t) dt} - g(y) - \frac{C_D(\rho, V, y) \frac{1}{2} \rho(y) A V^2}{M_o - \int_0^t \dot{m}(t) dt} \dots\dots\dots (1)$$

$$y'' = \frac{F}{M_o - \dot{m}t} - g \quad y(0.0) = 0.0 \quad \text{and} \quad y'(0.0) = V(0.0) = 0.0 \quad \dots\dots\dots (2)$$

Equations (1) and (2) both can be reduced to a system of two coupled initial-value ODEs by the procedure described below.

Consider the general n^{th} -order ODE:

$$y^{(n)} = f(t, y, y', y'', \dots, y^{(n-1)}) \dots\dots\dots (3)$$

$$\bar{y}(t_0) = \bar{y}_0 \quad \text{and} \quad \bar{y}^{(i)}(t_0) = \bar{y}_0^{(i)} \quad (i = 1, 2, \dots, n-1) \dots\dots\dots (4)$$

Equation (3) can be replaced by an equivalent system of n coupled first-order ODEs by defining n auxiliary variable. Thus,

$$\begin{aligned} y_1 &= y \\ y_2 &= y' = y_1' \\ y_3 &= y'' = y_2' \\ &\dots\dots\dots \dots\dots\dots (5.1-5.n) \end{aligned}$$

$$y_n = y^{(n-1)} = y_{n-1}'$$

Differentiating Eq. (5.n) gives

$$y_n' = y^{(n)} \dots\dots\dots (6)$$

Rearranging Eqs. (5.2) to (5.n) and substituting these results and Eq. (6) into Eq. (3) yields the following system of n coupled first-order ODEs:

$$\begin{aligned}
 y_1' &= y_2 & y_1(0) &= y_0 \\
 y_2' &= y_3 & y_2(0) &= y_0' \\
 &\dots\dots\dots & & \\
 y_{n-1}' &= y_n & y_{n-1}(0) &= y_0^{(n-2)} \dots\dots\dots (7.1-7.n) \\
 y_n' &= F(t, y_1, y_2, \dots, y_n) & y_n(0) &= y_0^{(n-1)}
 \end{aligned}$$

where Eq.(7.n) is the original n^{th} -order ODE, Eq. (3), expressed in terms of the auxiliary variables y_i ($i = 1, 2, \dots, n$).

The result is a system of n coupled first-order ODEs. This reduction can nearly always be done. Thus, the general features of a higher-order ODE are similar to the general features of a first-order ODE.

Example: Reduction of a second-order ODE to two coupled first-order ODEs

To illustrate the reduction of a higher-order ODE to a system of coupled first-order ODEs, let's reduce Eq. (2) to a system of two coupled first-order ODEs.

$$y'' = \frac{F}{M_o - \dot{m}t} - g \quad y(0.0) = 0.0 \quad \text{and} \quad y'(0.0) = V(0.0) = 0.0 \quad \dots\dots\dots (8)$$

Let $y' = V$. Then Eq. (8) reduces to the following pair of coupled first-order ODEs:

$$y' = V \quad y(0.0) = 0.0 \quad \dots\dots\dots (9)$$

$$V' = \frac{F}{M_o - \dot{m}t} - g \quad V(0.0) = 0.0 \quad \dots\dots\dots (10)$$

Equations (9) and (10) comprise a system of two coupled first-order ODEs for $y(t)$ and $V(t)$. And this system will be solved later in this chapter.

Systems of first-order ordinary differential equations

In many applications in engineering and science, systems of coupled first-order ODEs governing several dependent variables arise. The methods for solving a single first-order ODE can be used to solve systems of coupled first-order ODEs.

Consider the system of n coupled first-order ODEs:

$$\bar{y}'_i = \bar{f}_i(t, \bar{y}_1, \bar{y}_2, \dots, \bar{y}_n) \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (1)$$

$$\bar{y}'_i(0.0) = Y_i \quad (i = 1, 2, \dots, n) \quad \dots\dots\dots (2)$$

Each ODE in the system of ODEs can be solved by any of the methods developed for solving single ODEs. Care must be taken to ensure the proper coupling of the solutions. When predictor-corrector or multistep methods are used, each step must be applied to all the equations before proceeding to the next step. The step size must be the same for all the equations.

Example: Solution of two coupled first-order ODEs

Consider the system of two coupled linear first-order initial-value ODEs develop in the last example for the vertical motion of a rocket:

$$y' = V \quad y(0.0) = 0.0 \quad \dots\dots\dots (3)$$

$$V' = \frac{F}{M_o - \dot{m}t} - g \quad V(0.0) = 0.0 \quad \dots\dots\dots (4)$$

Where M_0 is the initial mass, \dot{m} is the mass explosion rate, and g is the acceleration of gravity. The exact solution of Eq.(4) is

$$V(t) = -\frac{T}{\dot{m}} \ln\left(1 - \frac{\dot{m}t}{M_0}\right) - gt \quad \dots\dots\dots (5)$$

Substituting Eq.(5) into Eq.(3) and integrating yields the exact solution of Eq.(3):

$$y(t) = \frac{M_0}{\dot{m}} \left(\frac{T}{\dot{m}}\right) \left(1 - \frac{\dot{m}t}{M_0}\right) \ln\left(1 - \frac{\dot{m}t}{M_0}\right) + \frac{Tt}{\dot{m}} - \frac{1}{2}gt^2 \quad \dots\dots\dots (6)$$

As an example, let $T = 10,000\text{N}$, $M_0 = 100.0\text{ kg}$, $\dot{m} = 5.0\text{kg/s}$, and $g = 9.8\text{m/s}^2$.

Equations (3) and (4) become

$$y' = f(t, y, V) = V \quad y(0.0) = 0.0 \quad \dots\dots\dots (7)$$

$$V' = g(t, y, V) = \frac{10,000.0}{100.0 - 5.0t} - 9.8 \quad V(0.0) = 0.0 \quad \dots\dots\dots (8)$$

Equations (5) and (6) become

$$V(t) = -1000\ln(1 - 0.05t) - 9.8t \quad \dots\dots\dots (9)$$

$$y(t) = 10,000(1 - 0.05t)\ln(1 - 0.05t) - 2000t - 4.9t^2 \quad \dots\dots\dots (10)$$

Let's solve this problem by the fourth order Runge-Kutta method, for $V(10.0)$ and $y(10.0)$ with $\Delta t = 1.0\text{ s}$. Let $\Delta y_i (i = 1, 2, 3, 4)$ denote the increment in $y(t)$ and $\Delta V_i (i = 1, 2, 3, 4)$ denote the increments in $V(t)$. Thus,

$$y_{n+1} = y_n + \frac{1}{6}(\Delta y_1 + 2\Delta y_2 + 2\Delta y_3 + \Delta y_4) \quad \dots\dots\dots (11)$$

$$V_{n+1} = V_n + \frac{1}{6}(\Delta V_1 + 2\Delta V_2 + 2\Delta V_3 + \Delta V_4) \quad \dots\dots\dots (12)$$

where $\Delta y_i (i = 1, 2, 3, 4)$ and $\Delta V_i (i = 1, 2, 3, 4)$ are given by

$$\Delta y_1 = \Delta t f(t_n, y_n, V_n) \quad \Delta V_1 = \Delta t g(t_n, y_n, V_n) \quad \dots\dots\dots (13.a)$$

$$\begin{aligned} \Delta y_2 &= \Delta t f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta y_1}{2}, V_n + \frac{\Delta V_1}{2}\right) \\ \Delta V_2 &= \Delta t g\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta y_1}{2}, V_n + \frac{\Delta V_1}{2}\right) \quad \dots\dots\dots (13.b) \end{aligned}$$

$$\begin{aligned} \Delta y_3 &= \Delta t f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta y_2}{2}, V_n + \frac{\Delta V_2}{2}\right) \\ \Delta V_3 &= \Delta t g\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta y_2}{2}, V_n + \frac{\Delta V_2}{2}\right) \quad \dots\dots\dots (13.c) \end{aligned}$$

$$\begin{aligned}\Delta y_4 &= \Delta t f(t_n + \Delta t, y_n + \Delta y_3, V_n + \Delta V_3) \\ \Delta V_4 &= \Delta t g(t_n + \Delta t, y_n + \Delta y_3, V_n + \Delta V_3) \dots\dots\dots (13.d)\end{aligned}$$

Due to the coupling, Δy_1 and ΔV_1 both must be computed before Δy_2 and ΔV_2 can be computed, and Δy_2 and ΔV_2 must be computed before Δy_3 and ΔV_3 can be computed, etc.

The derivative functions, $f(t, y, V)$ and $g(t, y, V)$ are given by Eqs. (7) and (8), respectively. Thus, Eq. (13) reduces to

$$\Delta y_1 = \Delta t(V_n) \quad \Delta V_1 = \Delta t \left(\frac{10,000.0}{100.0 - 5.0t_n} - 9.8 \right) \dots\dots\dots (14.a)$$

$$\Delta y_2 = \Delta t \left(V_n + \frac{\Delta V_1}{2} \right) \quad \Delta V_2 = \Delta t \left[\frac{10,000.0}{100.0 - 5.0(t_n + \Delta t/2)} - 9.8 \right] \dots\dots\dots (14.b)$$

$$\Delta y_3 = \Delta t \left(V_n + \frac{\Delta V_2}{2} \right) \quad \Delta V_3 = \Delta t \left[\frac{10,000.0}{100.0 - 5.0(t_n + \Delta t/2)} - 9.8 \right] \dots\dots\dots (14.c)$$

$$\Delta y_4 = \Delta t(V_n + \Delta V_3) \quad \Delta V_3 = \Delta t \left[\frac{10,000.0}{100.0 - 5.0(t_n + \Delta t)} - 9.8 \right] \dots\dots\dots (14.d)$$

Let $\Delta t = 1.0\text{s}$. For the first time step,

$$\Delta y_1 = 1.0(0.0) = 0.000000$$

$$\Delta V_1 = 1.0 \left[\frac{10,000.0}{100.0 - 5.0(0.0)} - 9.8 \right] = 90.200000$$

$$\Delta y_2 = 1.0 \left(0.0 + \frac{90.200000}{2} \right) = 45.100000$$

$$\Delta V_2 = 1.0 \left[\frac{10,000.0}{100.0 - 5.0(0.0 + 1.0/2)} - 9.8 \right] = 95.463158$$

$$\Delta y_3 = 1.0 \left(0.0 + \frac{95.463158}{2} \right) = 47.731579$$

$$\Delta V_3 = 1.0 \left[\frac{10,000.0}{100.0 - 5.0(0.0 + 1.0/2)} - 9.8 \right] = 95.463158$$

$$\Delta y_4 = 1.0(0.0 + 95.463158) = 95.463158$$

$$\Delta V_4 = 1.0 \left[\frac{10,000.0}{100.0 - 5.0(0.0 + 1.0)} - 9.8 \right] = 101.311111$$

Substituting these results into Eqs. (11) and (12) gives

$$y_{n+1} = 0.0 + \frac{1}{6}(0.0 + 2(45.100000 + 47.731579) + 95.463158) = 46.854386 \text{ m}$$

$$V_{n+1} = 0.0 + \frac{1}{6}(90.200000 + 2(95.463158 + 95.463158) + 101.311111) = 95.560624 \text{ m/s}$$

These results and the results for the subsequent time steps for $t = 2.0$ to 10.0 s are presented in the following table.

Table Solution of Two Coupled First-Order ODEs					
t_n	y_n	Δy_1	Δy_2	Δy_3	Δy_4
	V_n	ΔV_1	ΔV_2	ΔV_3	ΔV_4
t_{n+1}	y_{n+1}				
	V_{n+1}				
0.00	0.00000000	0.00000000	45.10000000	46.38205128	92.76410256
	0.00000000	90.20000000	92.76410256	92.76410256	95.46315789
1.00	45.95470085	92.78659469	140.51817364	141.94064875	191.09470280
	92.78659469	95.46315789	98.30810811	98.30810811	101.31111111
2.00	187.41229123	191.12104493	241.77660049	243.36390207	295.60675922
	191.12104493	101.31111111	104.48571429	104.48571429	107.84705882
3.00	430.25599278	295.63788278	349.56141219	351.34394338	407.05000399
	295.63788278	107.84705882	111.41212121	111.41212121	115.20000000
...
9.00	4450.68315419	1107.47425871	1193.48334962	1197.81235395	1288.15044919
	1107.47425871	172.01818182	180.67619048	180.67619048	190.20000000
10.00	5647.05250670				
	1288.29474933				

Numerical Analysis

DWE3214

PART-II: Numerical Solutions of ODE

Unit-7: Boundary Value Problem



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Unit-7: Boundary Value Problem

Introduction

A classic example of a boundary-value ODE is the general second-order ODE:

$$y'' + P(x, y)y' + Q(x, y)y = F(x) \quad y(x_1) = y_1 \quad \text{and} \quad y(x_2) = y_2$$

This equation applies to many problems in engineering and science.

Consider the constant cross-sectional area rod illustrated in the figure. Heat diffusion transfers energy along the rod and energy is transferred from the rod to the surroundings by convection. An energy balance on the differential control volume yields

$$\dot{q}(x) = \dot{q}(x + dx) + \dot{q}_c(x)$$

which can be written as

$$\dot{q}(x) = \dot{q}(x) + \frac{d}{dx} \left[\dot{q}(x) \right] dx + \dot{q}_c(x)$$

which yields

$$\frac{d}{dx} \left[\dot{q}(x) \right] dx + \dot{q}_c(x) = 0$$

Heat diffusion is governed by **Fourier law of conduction**, which states that

$$\dot{q}(x) = -kA \frac{dT}{dx}$$

where $\dot{q}(x)$ is the energy transfer rate (J/s), k is the thermal conductivity of the solid (J/s-m-K), A is the cross-sectional area of the rod (m^2), and dT/dx is the temperature gradient (K/m). Heat transfer by convection is governed by **Newton's law of cooling**:

$$\dot{q}_c(x) = hA(T - T_a)$$

where h is the empirical heat transfer coefficient (J/s- m^2 -K), A is the surface area of the rod ($A = P dx$, m^2), P is the perimeter of the rod (m), and T_a is the ambient temperature (K) (i.e., temperature of the surroundings). From the last three equations we can see that

$$\frac{d}{dx} \left(-kA \frac{dT}{dx} \right) dx + h(P dx)(T - T_a) = 0$$

For constant k , A , and P , the last equation yields

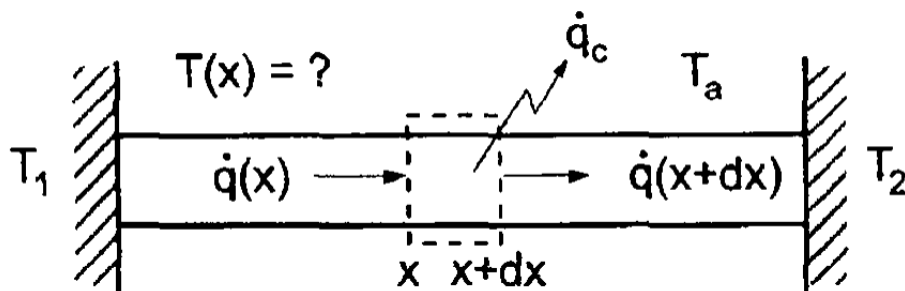
$$\frac{d^2 T}{dx^2} = \frac{hP}{kA} (T - T_a) = 0$$

which can be written as

$$T'' - \alpha^2 T = -\alpha^2 T_a \quad \text{where} \quad \alpha^2 = \frac{hP}{kA}$$

which is a linear second-order boundary-value **ODE**. The solution of this equation is the function $T(x)$, which describes the temperature distribution in the rod corresponding to the boundary conditions

$$T(x_1) = T_1 \quad \text{and} \quad T(x_2) = T_2$$



Steady heat conduction in a rod

An example of a higher-order boundary-value **ODE** is given by the fourth-order **ODE** governing the deflection of a laterally loaded symmetrical beam. The physical system is illustrated in Figure (5.8). Bending takes place in the plane of symmetry, which causes deflections in the beam. The neutral axis of the beam is the axis along which the fibers do not undergo strain during bending. When no load is applied (i.e., neglecting the weight of the beam itself), the neutral axis is coincident with the x -axis. When a distributed load $q(x)$ is applied, the beam deflects, and the neutral axis is displaced, as illustrated by the dashed line in Figure (II.8). The shape of the neutral axis is called the deflection curve.

As shown in many strength of materials books (e.g. Timoshenko, 1955), the differential equation of the deflection curve is

$$EI(x) \frac{d^2 y}{dx^2} = -M(x)$$

where E is the modulus of elasticity of the beam material, $I(x)$ is the moment of inertia of the beam cross-section, which can vary along the length of the beam, and $M(x)$ is the bending moment due to transverse forces on the beam, which can vary along the length of the beam. The moment $M(x)$ is related to the shearing forces $V(x)$ acting on each cross-section of the beam as follows:

$$\frac{dM(x)}{dx} = V(x)$$

The shearing forces $V(x)$ is related to the distributed load $q(x)$ as follows:

$$\frac{dV(x)}{dx} = -q(x)$$

Combining the three equations yields the differential equation for the beam deflection curve:

$$EI(x) \frac{d^4 y}{dx^4} = q(x)$$

This equation requires four boundary conditions. For a horizontal beam of length L ,

For a supported beam at both ends (any kind of rigid support):

$$y(0) = y(L) = 0.0$$

For a beam fixed (i.e., clamped) at both ends:

$$y'(0) = y'(L) = 0.0$$

For a beam pinned (i.e., hinged) at both ends:

$$y''(0) = y''(L) = 0.0$$

For a beam cantilevered (i.e., free) at either ends:

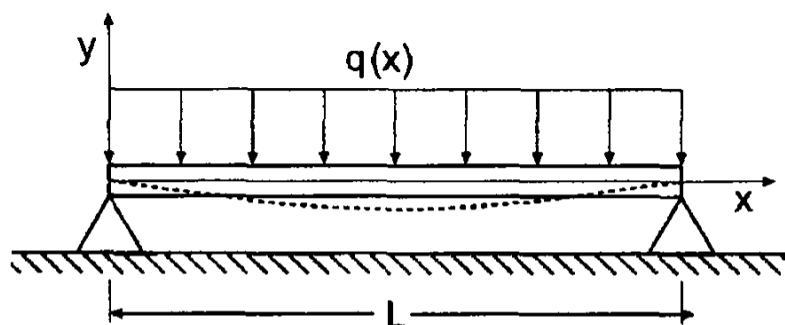
$$y'''(0) = 0.0 \quad \text{or} \quad y'''(L) = 0.0$$

Any two combinations of these four boundary conditions can be specified at each end.

The last equation is a linear example of the general nonlinear fourth-order boundary-value **ODE**:

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

which requires four boundary conditions at the boundaries of the closed physical domain.



$$EI(x) \frac{d^4 y}{dx^4} = q(x)$$

$$y(0) = 0, \quad y''(0) = 0; \quad y(L) = 0, \quad \text{and} \quad y''(L) = 0$$

Deflection of a beam

The Equilibrium (Boundary-Value) Method

The solution of boundary-value problems by the equilibrium (boundary-value) method is accomplished by the following steps:

1. **Discretizing** the continuous solution domain into a discrete finite difference grid.
2. **Approximating** the exact derivatives in the boundary-value ODE by algebraic finite difference approximations (FDAs).
3. **Substituting** the FDAs into the ODE to obtain an algebraic finite difference equation (FDE).
4. **Solving** the resulting system of algebraic FDEs.

When the finite difference equation is applied at every point in the discrete finite difference grid, a system of coupled finite difference equations results, which must be solved simultaneously, thus relaxing the entire solution, including the boundary points, simultaneously.

When solving boundary-value problems by the equilibrium method, consistency, order, and convergence of the solution method must be considered. Stability is not an issue, since a relaxation procedure, not a marching procedure is employed. Consistency and order are determined by a Taylor series consistency analysis, which is discussed earlier for marching methods. The same procedure is applicable to relaxation methods. Convergence is guaranteed for consistent finite difference approximations of a boundary-value ODE, as long as the system of FDEs can be solved. In principle, this can always be accomplished by direct solution methods, such as Gauss elimination.

The second-order boundary-value ODE

Consider the linear, variable coefficient, second-order boundary-value problem with Dirichlet boundary conditions:

$$\bar{y}'' + P(x)\bar{y}' + Q(x)\bar{y} = F(x) \quad \bar{y}(x_1) = \bar{y}_1 \text{ and } \bar{y}(x_2) = \bar{y}_2 \quad \dots\dots\dots (1)$$

The discrete finite difference grid for solving Eq. (1) by the equilibrium method is illustrated in the next figure. Recall the second order centered-difference approximations of $\bar{y}'|_i$ and $\bar{y}''|_i$ at grid point i .

$$\bar{y}'|_i = \frac{\bar{y}_{i+1} - \bar{y}_{i-1}}{2\Delta x} + O(\Delta x^2) \quad \dots\dots\dots (2)$$

$$\bar{y}''|_i = \frac{\bar{y}_{i+1} - 2\bar{y}_i + \bar{y}_{i-1}}{\Delta x^2} + O(\Delta x^2) \quad \dots\dots\dots (3)$$

Substituting Eqs. (2) and (3) into Eq. (1) and evaluating the coefficients $P(x)$ and $Q(x)$ at grid point i yields

$$\bar{y}''|_i = \frac{\bar{y}_{i+1} - 2\bar{y}_i + \bar{y}_{i-1}}{\Delta x^2} + O(\Delta x^2) + P_i \left[\frac{\bar{y}_{i+1} - \bar{y}_{i-1}}{2\Delta x} + O(\Delta x^2) \right] + Q_i \bar{y}_i = F_i \quad \dots\dots\dots (4)$$

All the approximations in Eq. (4) are $O(\Delta x^2)$. Multiplying Eq. (4) through by Δx^2 , gathering terms, and truncating the remainder terms yields:

$$\left(1 - \frac{\Delta x}{2} P_i\right) y_{i-1} + (-2 + \Delta x^2 Q_i) y_i + \left(1 + \frac{\Delta x}{2} P_i\right) y_{i+1} = \Delta x^2 F_i \quad \dots\dots\dots (5)$$

Applying Eq. (5) at each point in a discrete finite difference grid yields a tridiagonal system of FDEs, which can be solved by the Thomas algorithm.

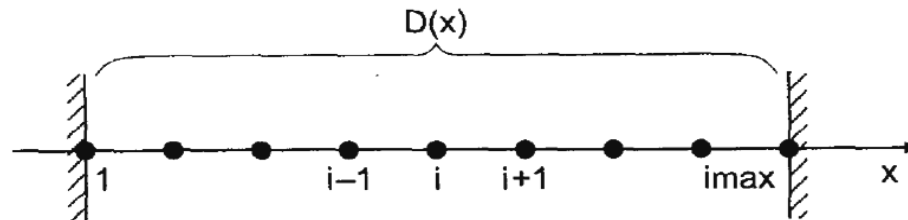


Figure Solution domain $D(x)$ and finite difference grid.

Example: The second-order equilibrium method

Let's solve the heat transfer problem by the second-order equilibrium method. The boundary-value ODE is:

$$T'' - \alpha^2 T = -\alpha^2 T_a \quad T(0.0) = 0.0\text{C} \text{ and } T(1.0) = 100.0\text{C} \quad \dots\dots\dots (6)$$

Replacing the T'' by the second order centered-difference approximation, Eq. (3), and evaluating all the terms at grid point i gives

$$\frac{\bar{T}_{i+1} - 2\bar{T}_i + \bar{T}_{i-1}}{\Delta x^2} + O(\Delta x^2) - \alpha^2 \bar{T}_i = -\alpha^2 T_a \quad \dots\dots\dots (7)$$

Multiplying through by Δx^2 , gathering terms, and truncating the remainder term yield the FDE:

$$T_{i-1} - (2 + \alpha^2 \Delta x^2) T_i + T_{i+1} = -\alpha^2 \Delta x^2 T_a \quad \dots\dots\dots (8)$$

Let $\alpha^2 = 16.0 \text{ cm}^{-2}$, $T_a = 0.0\text{C}$, and $\Delta x = 0.25 \text{ cm}$. Then Eq. (8) becomes

$$T_{i-1} - 3T_i + T_{i+1} = 0 \quad \dots\dots\dots (9)$$

Applying Eq. (9) at the three interior grid points, $x = 0.25, 0.5$, and 0.75 cm, gives

$$\begin{aligned} x = 0.25: \quad T_1 - 3T_2 + T_3 &= 0.0 & T_1 = \bar{T}_1 &= 0.0 \\ x = 0.50: \quad T_2 - 3T_3 + T_4 &= 0.0 & & \dots\dots\dots (10.a-c) \\ x = 0.75: \quad T_3 - 3T_4 + T_5 &= 0.0 & T_5 = \bar{T}_5 &= 100.0 \end{aligned}$$

Transferring T_1 and T_5 to the right-hand side of Eqs. (10.a) and (10.c), respectively, yields the following tridiagonal system of FDEs:

$$\begin{bmatrix} -3.0 & 1.0 & 0.0 \\ 1.0 & -3.0 & 1.0 \\ 0.0 & 1.0 & -3.0 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \\ -100.0 \end{bmatrix} \quad \dots\dots\dots (11)$$

Solving Eq. (11) by the Thomas algorithm yields the results presented in the next table. The exact solution and the errors are presented for comparison.

Table Solution by the Equilibrium Method for $\Delta x = 0.25$ cm			
x , cm	$T(x)$, C	$\bar{T}(x)$, C	Error(x), C
0.00	0.000000	0.000000	
0.25	4.761905	4.306357	0.455548
0.50	14.285714	13.290111	0.995603
0.75	38.095238	36.709070	1.386168
1.00	100.000000	100.000000	

Let's repeat the solution for $\Delta x = 0.125$ cm. In this case Eq. (8) becomes

$$T_{i-1} - 2.25T_i + T_{i+1} = 0 \quad \dots\dots\dots (12)$$

Applying Eq. (12) at the seven interior grid points gives

$$\begin{aligned}
 x = 0.125: & \quad T_1 - 2.25T_2 + T_3 = 0.0 & \quad T_1 = \bar{T}_1 = 0.0 \\
 x = 0.250: & \quad T_2 - 2.25T_3 + T_4 = 0.0 \\
 x = 0.375: & \quad T_3 - 2.25T_4 + T_5 = 0.0 \\
 x = 0.500: & \quad T_4 - 2.25T_5 + T_6 = 0.0 & \quad \dots\dots\dots (13.a-g) \\
 x = 0.625: & \quad T_5 - 2.25T_6 + T_7 = 0.0 \\
 x = 0.750: & \quad T_6 - 2.25T_7 + T_8 = 0.0 \\
 x = 0.875: & \quad T_7 - 2.25T_8 + T_9 = 0.0 & \quad T_9 = \bar{T}_9 = 100.0
 \end{aligned}$$

Transferring T_1 and T_9 to the right-hand sides of Eqs. (13.a) and (13.g), respectively, yields a tridiagonal system of equations. That tridiagonal system of equations is solved by the Thomas algorithm in Chapter 2. The results are presented in the next table.

Table Solution by the Equilibrium Method for $\Delta x = 0.125$ cm

x , cm	$T(x)$, C	$\bar{T}(x)$, C	Error(x), C
0.000	0.000000	0.000000	
0.125	1.966751	1.909479	0.057272
0.250	4.425190	4.306357	0.118833
0.375	7.989926	7.802440	0.187486
0.500	13.552144	13.290111	0.262033
0.625	22.502398	22.170109	0.332288
0.750	37.078251	36.709070	0.369181
0.875	60.923667	60.618093	0.305575
1.000	100.000000	100.000000	

The Euclidean norm of the errors for $\Delta x = 0.25$ cm is 1.766412 C. The Euclidean norm of the errors for $\Delta x = 0.125$ cm at the three common grid points is 0.468057 C. The ratio of the norms is 3.77. The ratios of the individual errors at the three common points in the two grids are 3.83, 3.80, and 3.75. Both results demonstrate that the method is second order.

The errors of the second-order equilibrium method are about 40 percent of the magnitude of the errors of the second order shooting method. The errors in both cases can be decreased by using a smaller step size or a higher-order method. The errors are illustrated in the next figure, which also presents the errors of the compact three-point fourth-order equilibrium method presented later, as well as the errors from extrapolation of the second-order method, which is presented later also. For the fourth-order method, the Euclidean norms of the errors at the three common grid points in the two grids are 0.092448 C and 0.005919 C, respectively. The ratio of the norms is 15.62, which demonstrates the fourth-order behavior of the method.

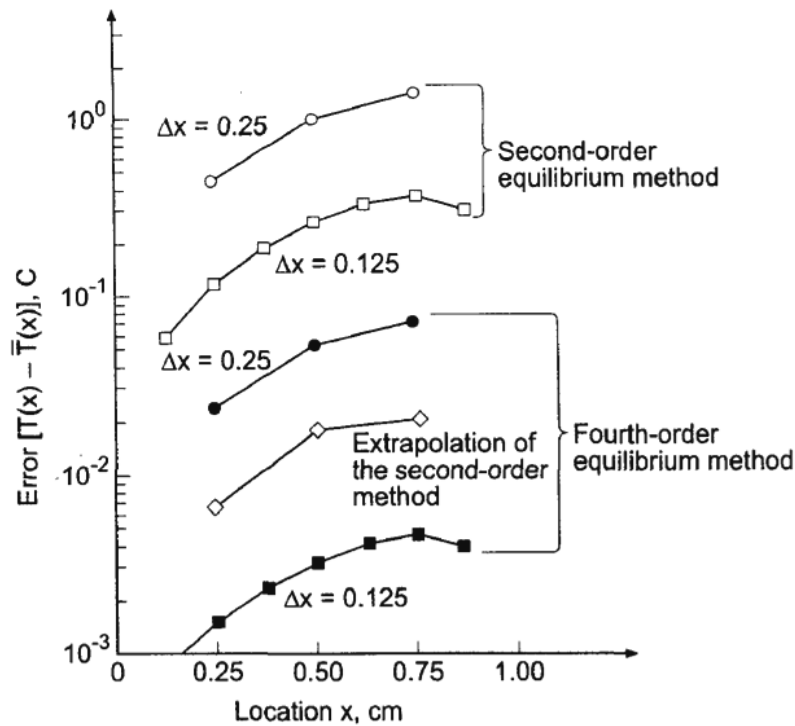


Figure Errors in the solution by the equilibrium method.

Extrapolation

The second-order results obtained in the last example by the equilibrium method can be extrapolated by the same procedure presented for the shooting method.

Example: The second-order equilibrium method by extrapolation

Let's apply extrapolation to the results obtained in the last example. Those results were presented in the last two tables. The results at the three common grid points in the two grids are summarized in the next table. For these results, $R = 0.25/0.125 = 2.0$.

$$IV = MAV + \frac{1}{2^2 - 1} (MAV - LAV) = \frac{4MAV - LAV}{3}$$

The results obtained by applying the above equation is also presented in the next table and the last figure. The Euclidean norm of these errors is 0.035514 C, which is 13.18 times smaller than the Euclidean norm of the errors of the second-order equilibrium method without extrapolation.

Table Solution by Extrapolation of the Second-Order Equilibrium Method Results

x , cm	$T(\text{LAV})$, C	$T(\text{MAV})$, C	$T(\text{IV})$, C	$\bar{T}(x)$, C	Error(x), C
0.00	0.000000	0.000000	0.000000	0.000000	
0.25	4.761905	4.425190	4.312952	4.306357	0.006595
0.50	14.285714	13.552144	13.307621	13.290111	0.017510
0.75	38.095238	37.078251	36.739255	36.709070	0.030185
1.00	100.000000	100.000000	100.000000	100.000000	

Higher-order boundary-value ODEs

Consider the general nonlinear fourth-order boundary-value problem

$$y'''' = f(x, y, y', y'', y''') \quad \dots\dots\dots (1)$$

Eq. (1) is fourth-order, four boundary conditions are required. At least one boundary condition must be specified on each boundary of the closed solution domain. The two remaining boundary conditions can be specified on the same boundary, or one on each boundary. These boundary conditions can depend on y , y' , y'' , or y''' .

A finite difference approximation (FDA) must be developed for every derivative in Eq. (1). All the FDAs should be the same order. Second-order centered-difference FDAs can be developed for all four derivatives in Eq. (1). The first and second derivatives involve three grid points, points $i-1$ to $i+1$. The third and fourth derivatives involve five grid points, points $i-2$ to $i+2$. Consequently, the resulting finite difference equation (FDE) involves five grid points. Applying this FDE at each point in a finite difference grid yields a penta-diagonal system of FDEs, which can be solved by an algorithm similar to the Thomas algorithm for tridiagonal system of FDEs.

Example: A fourth-order ODE by the second-order equilibrium method

Let's solve the deflection problem for a laterally loaded symmetrical beam, expressed in the form of

$$y'''' = \frac{q(x)}{EI(x)} \quad \dots\dots\dots (2)$$

where E is the modulus of elasticity of the beam material, $I(x)$ is the moment of inertia of the beam cross section. And $q(x)$ is the distributed load on the beam. Let's consider a rectangular cross section beam, for which $I = wh^3/12$, where w is the width of the beam and h is the height of the beam, and a uniform distributed load $q(x) = q = \text{constant}$.

The ends of the beam are at the same elevation, which can be chosen as $y = 0.0$. Thus, $y(0.0) = y(L) = 0.0$, where L is the length of the beam. One additional boundary condition is required at each end of the beam. If the beam is fixed, then y' is specified. If the beam is pinned (i.e., clamped), then $y'' = 0.0$. If the beam is free (i.e. cantilevered), then $y''' = 0.0$. Let's assume that both ends of the beam are pinned. Thus, $y''(0.0) = y''(L) = 0.0$.

Thus, the problem to be solved, including the boundary conditions, is given by

$$y'''' = \frac{q}{EI} \quad y(0.0) = y''(0.0) = y(L) = y''(L) = 0.0 \quad \dots\dots\dots (3)$$

The exact solution of Eq. (3) is

$$y(x) = \frac{qx^4}{24EI} - \frac{qLx^3}{12EI} + \frac{qL^3x}{24EI} \quad \dots\dots\dots (4)$$

As an example, let $q = -2000.0$ N/m, $L = 5.0$ m, $w = 5.0$ cm, $h = 10.0$ cm, and $E = 90 \times 10^9$ N/m². Then Eqs. (3) and (4) become

$$y'''' = -0.0024 \quad y(0.0) = y''(0.0) = y(5.0) = y''(5.0) = 0.0 \quad \dots\dots\dots (5)$$

$$y(x) = 0.000100x^4 - 0.001000x^3 + 0.012500x \quad \dots\dots\dots (6)$$

Let's solve this problem using a second order centered-difference approximation for y'''' .

Write Taylor series for $\bar{y}_{i\pm1}$ and $\bar{y}_{i\pm2}$ with base point i :

$$\begin{aligned} \bar{y}_{i\pm1} = & \bar{y}_i \pm \bar{y}'_i \Delta x + \frac{1}{2} \bar{y}''_i \Delta x^2 \pm \frac{1}{6} \bar{y}'''_i \Delta x^3 + \frac{1}{24} \bar{y}''''_i \Delta x^4 \\ & \pm \frac{1}{120} \bar{y}^{(v)}_i \Delta x^5 + \frac{1}{720} \bar{y}^{(vi)}_i \Delta x^6 \pm \dots \end{aligned} \quad \dots\dots\dots (7)$$

$$\begin{aligned} \bar{y}_{i\pm2} = & \bar{y}_i \pm 2\bar{y}'_i \Delta x + \frac{4}{2} \bar{y}''_i \Delta x^2 \pm \frac{8}{6} \bar{y}'''_i \Delta x^3 + \frac{16}{24} \bar{y}''''_i \Delta x^4 \\ & \pm \frac{32}{120} \bar{y}^{(v)}_i \Delta x^5 + \frac{64}{720} \bar{y}^{(vi)}_i \Delta x^6 \pm \dots \end{aligned} \quad \dots\dots\dots (8)$$

Adding \bar{y}_{i+1} and \bar{y}_{i-1} gives

$$(\bar{y}_{i+1} + \bar{y}_{i-1}) = 2\bar{y}_i + \frac{2}{2} \bar{y}''_i \Delta x^2 + \frac{2}{24} \bar{y}''''_i \Delta x^4 + \frac{2}{720} \bar{y}^{(vi)}_i \Delta x^6 \pm \dots \quad \dots\dots\dots (9)$$

Adding \bar{y}_{i+2} and \bar{y}_{i-2} gives

$$(\bar{y}_{i+2} + \bar{y}_{i-2}) = 2\bar{y}_i + \frac{8}{2} \bar{y}''|_i \Delta x^2 + \frac{32}{24} \bar{y}'''|_i \Delta x^4 + \frac{128}{720} \bar{y}^{(vi)}|_i \Delta x^6 \pm \dots \quad (10)$$

Subtracting $4(\bar{y}_{i+1} + \bar{y}_{i-1})$ from $(\bar{y}_{i+2} + \bar{y}_{i-2})$ yields

$$(\bar{y}_{i+2} + \bar{y}_{i-2}) - 4(\bar{y}_{i+1} + \bar{y}_{i-1}) = 6\bar{y}_i + \bar{y}'''|_i \Delta x^4 + \frac{1}{6} \bar{y}^{(vi)}|_i \Delta x^6 \pm \dots \quad (11)$$

Solving Eq. (11) for $\bar{y}'''|_i$ yields

$$\bar{y}'''|_i = \frac{\bar{y}_{i-2} - 4\bar{y}_{i-1} + 6\bar{y}_i - 4\bar{y}_{i+1} + \bar{y}_{i+2}}{\Delta x^4} - \frac{1}{6} \bar{y}^{(vi)}(\xi) \Delta x^2 \quad (12)$$

where $x_{i-2} \leq \xi \leq x_{i+2}$. Truncating the remainder term yields a second order centered-difference approximation for $\bar{y}'''|_i$:

$$\bar{y}'''|_i = \frac{\bar{y}_{i-2} - 4\bar{y}_{i-1} + 6\bar{y}_i - 4\bar{y}_{i+1} + \bar{y}_{i+2}}{\Delta x^4} \quad (13)$$

Substituting Eq. (13) into Eq. (5) yields

$$y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2} = -0.0024 \Delta x^4 \quad (14)$$

Let $\Delta x = 1.0$ m. Applying Eq. (14) at the four interior points illustrated in the next figure gives

$$\begin{aligned} x = 1.0: & \quad y_A - 4y_1 + 6y_2 - 4y_3 + y_4 = -0.0024 \\ x = 2.0: & \quad y_1 - 4y_2 + 6y_3 - 4y_4 + y_5 = -0.0024 \\ x = 3.0: & \quad y_2 - 4y_3 + 6y_4 - 4y_5 + y_6 = -0.0024 \quad (15.a-d) \\ x = 4.0: & \quad y_3 - 4y_4 + 6y_5 - 4y_6 + y_B = -0.0024 \end{aligned}$$

Note that y_1 in Eq. (15.a-b) and y_6 in Eq. (15.c-d) are zero. Grid points A and B are outside the physical domain. Thus, y_A and y_B are unknown. These values are determined by applying the boundary conditions $\bar{y}''(0.0) = \bar{y}''(L) = 0.0$. Applying the second derivative FDA formula at grid points 1 and 6 gives

$$y''|_1 = \frac{y_2 - 2y_1 + y_A}{\Delta x^2} = 0.0$$

$$y''|_6 = \frac{y_B - 2y_6 + y_5}{\Delta x^2} = 0.0 \quad \dots\dots\dots (16.a-b)$$

Solving Eq. (16) for y_A and y_B , with $y_1 = y_6 = 0.0$, gives

$$y_A = -y_2 \quad \text{and} \quad y_B = -y_5 \quad \dots\dots\dots (17)$$

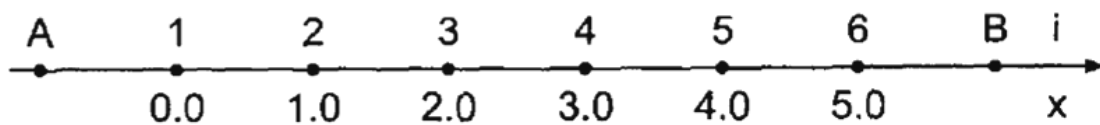


Figure Finite difference grid for the beam.

Substituting these values into Eqs. (15.a) and (15.d), respectively, and rearranging those two equations yields the following system equation:

$$5y_2 - 4y_3 + y_4 = -0.0024$$

$$-4y_2 + 6y_3 - 4y_4 + y_5 = -0.0024$$

$$y_2 - 4y_3 + 6y_4 - 4y_5 = -0.0024 \quad \dots\dots\dots (18.a-d)$$

$$y_3 - 4y_4 + 5y_5 = -0.0024$$

Expressing Eq. (18) in matrix form yields

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} -0.0024 \\ -0.0024 \\ -0.0024 \\ -0.0024 \end{bmatrix} \quad \dots\dots\dots (19)$$

Although it is not readily apparent, Eq. (19) is a penta-diagonal matrix, which can be solved very efficiently by a modified Gauss elimination algorithm similar to the Thomas algorithm for tridiagonal matrices. Solving Eq. (19) yields the presented in the next table. The exact solution and the errors are presented for comparison.

Table Solution by the Equilibrium Method
with $\Delta x = 1.0$ m

$x, \text{ m}$	$y(x), \text{ m}$	$\bar{y}(x), \text{ m}$	Error(x), m
0.00	0.000000	0.000000	'
1.00	0.012000	0.011600	0.000400
2.00	0.019200	0.018600	0.000600
3.00	0.019200	0.018600	0.000600
4.00	0.012000	0.011600	0.000400
5.00	0.000000	0.000000	

Let's repeat the solution for $\Delta x = 0.5$ m. In this case, Eq. (19) becomes

$$\begin{bmatrix}
 5 & -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 \\
 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 5
 \end{bmatrix}
 \begin{bmatrix}
 y_2 \\
 y_3 \\
 y_4 \\
 y_5 \\
 y_6 \\
 y_7 \\
 y_8 \\
 y_9 \\
 y_{10}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150 \\
 0.000150
 \end{bmatrix}
 \dots\dots\dots (20)$$

The penta-diagonal structure of Eq. (20) is readily apparent. Solving Eq. (20) yields the results presented in the next table.

The Euclidean norm of the errors for $\Delta x = 1.0$ m is 0.123456 m. The Euclidean norm of the errors for $\Delta x = 0.5$ m at the four common grid points is 0.012345 m. the ratio of the norms is 3.99. The ratios of the individual errors at the four common grid points in the two grids are 3.96, 3.97, 3.98, and 3.99. Both results demonstrate that the method is second order.

Table Solution by the Equilibrium Method
with $\Delta x = 0.5$ m

x , m	$y(x)$, m	$\bar{y}(x)$, m	Error(x), m
0.0	0.000000	0.000000	
0.5	0.006188	0.006131	0.000056
1.0	0.011700	0.011600	0.000100
1.5	0.016013	0.015881	0.000131
2.0	0.018750	0.018600	0.000150
2.5	0.019688	0.019531	0.000156
3.0	0.018750	0.018600	0.000150
3.5	0.016013	0.015881	0.000131
4.0	0.011700	0.011600	0.000100
4.5	0.006188	0.006131	0.000056
5.0	0.000000	0.000000	

Applications of FDM to determinate beams

Since moments in the determinate beams can be obtained by equilibrium method the following differential equation may be used:

$$\frac{d^2 y}{dx^2} = \frac{M}{EI}$$

In numerical representation

$$\frac{1}{h^2} \left\{ \begin{array}{ccc} \bigcirc 1 & \bigcirc \cdot & \bigcirc 1 \end{array} \right\} (y_i) = \frac{M}{EI}$$

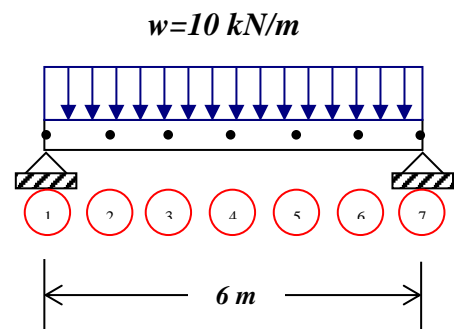
by using this stencil, the deflection of beam can be found at selected nodes.

Example: For the beam shown in the figure, find the deflection at nodes. Take $h = 1$ m.

Solution:

$$\begin{aligned} M &= \frac{wL}{2}(x) - \frac{wx^2}{2} \\ &= \frac{10 \times 6}{2}(x) - \frac{10(x^2)}{2} = 30x - 5x^2 \end{aligned}$$

$$(y_{i-1} - 2y_i + y_{i+1}) = \frac{1}{EI} (30x_i - 5x_i^2) \cdot h^2$$



$$y_1 - 2y_2 + y_3 = \frac{25}{EI}$$

$$y_2 - 2y_3 + y_4 = \frac{40}{EI}$$

$$y_3 - 2y_4 + y_5 = \frac{45}{EI}$$

$$y_4 - 2y_5 + y_6 = \frac{40}{EI}$$

$$y_5 - 2y_6 + y_7 = \frac{25}{EI}$$

Five Eqs but 7 unknowns then we need 2 boundary conditions:

$$y_1 = 0 \quad \text{and} \quad y_7 = 0$$

Solving the system of linear algebraic equations yields

$$\begin{Bmatrix} y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{Bmatrix} = -\frac{1}{EI} \begin{Bmatrix} 87.5 \\ 150 \\ 172.5 \\ 150 \\ 87.5 \end{Bmatrix}$$

Exact solution:

$$M = 30x - 5x^2 = EI \frac{d^2 y}{dx^2}$$

$$\therefore EI \cdot y = 5x^3 - \frac{5}{12}x^4 + c_1x + c_2 \quad \text{at } x=0 \quad y=0 \Rightarrow c_2 = 0$$

$$\text{at } x=6 \quad y=0 \Rightarrow c_1 = -90$$

$$\therefore EI \cdot y = 5x^3 - \frac{5}{12}x^4 - 90x$$

x (m)	y (Analytically)	y (Numerically)
1	-85.42/EI	-87.5/EI
2	-146.67/EI	-150/EI
3	-168.75/EI	-172.5/EI
4	-146.67/EI	-150/EI
5	-85.42/EI	-87.5/EI

The difference in maximum deflection is $\frac{172.5 - 168.75}{168.75} \times 100\% = 2.2\%$

Q- If $h = 0.5$ m then the difference will be small?!

Applications of FDM to indeterminate beams

For the indeterminate beams, the moments are unknowns, so the differential equation which is applicable will be:

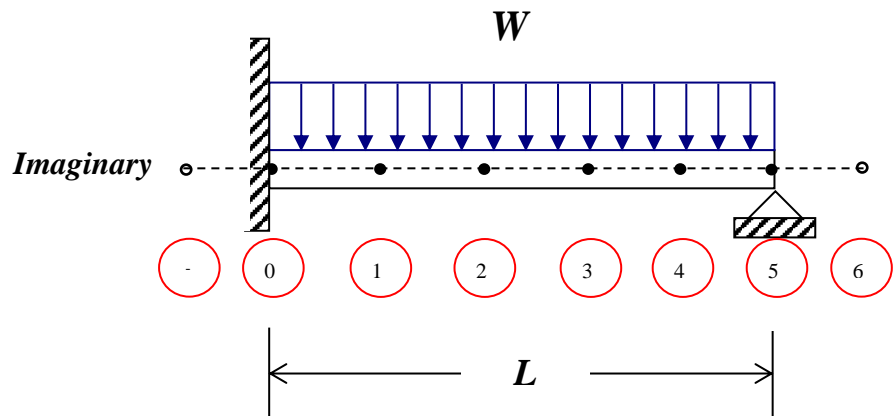
$$EI \frac{d^4 y}{dx^4} = W$$

The stencil representation is: -

$$\left\{ \begin{array}{ccccc} \textcircled{1} & \textcircled{-} & \textcircled{\textcircled{6}} & \textcircled{-} & \textcircled{1} \end{array} \right\} (y_i) = \frac{Wh^4}{EI}$$

Example: Find the deflection curve for the beam shown in the Fig below using $h=L/5$?

Solution:



$$(y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}) = \frac{Wh^4}{EI}$$

$$@ 1: y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 = \frac{Wh^4}{EI}$$

$$@ 2: y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 = \frac{Wh^4}{EI}$$

$$@ 3: y_1 - 4y_2 + 6y_3 - 4y_4 + y_5 = \frac{Wh^4}{EI}$$

$$@ 4: y_2 - 4y_3 + 6y_4 - 4y_5 + y_6 = \frac{Wh^4}{EI}$$

4-equations with 8-unknowns \rightarrow 4-boundary conditions needed:

Boundary Conditions:

1. $y_0 = 0$

2. $y_5 = 0$

3. at point 0 $\theta = \frac{dy}{dx} = 0 \Rightarrow \frac{y_1 - y_{-1}}{2h} = 0 \Rightarrow y_1 = y_{-1}$

4. at point 5 $M = EI \frac{d^2 y}{dx^2} = 0 \Rightarrow \frac{y_4 - 2y_5 + y_6}{h^2} = 0 \Rightarrow y_4 = -y_6$

If we substituted these 4-BC's into the above equations we get:

$$\begin{bmatrix} 7 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0.0768 \\ 0.0768 \\ 0.0768 \\ 0.0768 \end{bmatrix} \frac{WL^4}{48EI}$$

Solving the system yields:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0.1129412 \\ 0.2529882 \\ 0.2981647 \\ 0.2032914 \end{bmatrix} \frac{WL^4}{48EI}$$

The exact solution is:

$$y = \frac{WL^4}{48EI} \left[2 \left(\frac{x}{L} \right)^4 - 5 \left(\frac{x}{L} \right)^3 + 3 \left(\frac{x}{L} \right)^2 \right]$$

Substituting $x = 0.2L, 0.4L, 0.6L, 0.8L$ gives:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0.0832 \\ 0.2112 \\ 0.2592 \\ 0.1792 \end{bmatrix} \frac{WL^4}{48EI} \quad \text{The Error ranging from 15-30 \%}$$

If 10 divisions ($x=0.1L$) is chosen, then the numerical solution is:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0.029552 \\ 0.0906985 \\ 0.1607284 \\ 0.221731 \\ 0.260597 \end{bmatrix} \frac{WL^4}{48EI}$$
$$\begin{bmatrix} y_6 \\ y_7 \\ y_8 \\ y_9 \end{bmatrix} = \begin{bmatrix} 0.269015 \\ 0.243475 \\ 0.185266 \\ 0.100477 \end{bmatrix} \frac{WL^4}{48EI}$$

In this case the Max. Error is 9%

Approximate Eigenproblems

Eigenvalues of homogenous boundary-value problems can also be obtained by numerical methods. In this approach, the boundary-value ODE is approximated by a system of finite difference equations, and the values of the unknown parameter (i.e., eigenvalues) which satisfy the system of FDEs are determined. These values are approximations of the exact eigenvalues of the boundary-value problem.

Example: Approximation of eigenvalues by the equilibrium method.

Let's solve Eq. (1) for its eigenvalues by the finite difference method. Choose an equally spaced grid with four interior points, as illustrated in the next figure. Approximate \bar{y}'' with the second order centered-difference approximation. The corresponding finite difference equation is:

$$\frac{\bar{y}_{i+1} - 2\bar{y}_i + \bar{y}_{i-1}}{\Delta x^2} + 0(\Delta x^2) + k^2 \bar{y}_i = 0 \quad \dots\dots\dots (7)$$

Multiplying by Δx^2 , truncating the remainder term, and rearranging gives the FDE:

$$y_{i-1} - (2 - \Delta x^2 k^2) y_i + y_{i+1} = 0 \quad \dots\dots\dots (8)$$

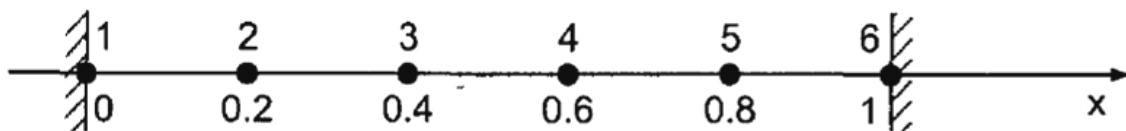


Figure Finite difference grid for the eigenproblem.

Apply the FDE, with $\Delta x = 0.2$, at the four interior points:

$$\begin{aligned} x = 0.2: & \quad y_1 - (2 - 0.04k^2)y_2 + y_3 = 0 & y_1 = 0 \\ x = 0.4: & \quad y_2 - (2 - 0.04k^2)y_3 + y_4 = 0 \\ x = 0.6: & \quad y_3 - (2 - 0.04k^2)y_4 + y_5 = 0 & \dots\dots\dots (9.a-d) \\ x = 0.8: & \quad y_4 - (2 - 0.04k^2)y_5 + y_6 = 0 & y_6 = 0 \end{aligned}$$

Writing Eq. (9) in matrix form gives

$$\begin{bmatrix} (2-0.04k^2) & -1 & 0 & 0 \\ -1 & (2-0.04k^2) & -1 & 0 \\ 0 & -1 & (2-0.04k^2) & -1 \\ 0 & 0 & -1 & (2-0.04k^2) \end{bmatrix} [y_i] = 0 \dots\dots\dots (10)$$

Which can be expressed as

$$(A - \lambda I)y = 0 \dots\dots\dots (11)$$

where $\lambda = 0.04k^2$ and A is defined as

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \dots\dots\dots (12)$$

This is a classical eigenproblem. The characteristic equation is given by

$$\det(A - \lambda I) = 0 \dots\dots\dots (13)$$

Define $Z = (2 - 0.04k^2)$. The characteristic equation is determined by expanding the determinant $|A - \lambda I| = 0$, which gives

$$Z^4 - 3Z^2 + 1 = 0 \dots\dots\dots (14)$$

which is quadratic in Z^2 . Solving Eq. (14) by the quadratic formula yields

$$Z = (2 - 0.04k^2) = \pm 1.618 \dots \pm 0.618 \dots \dots\dots (15)$$

The values of Z , k , $k(\text{exact})$, and percent error are presented in the next table.

The first eigenvalue is reasonably accurate. The higher-order eigenvalues become less and less accurate. To improve the accuracy of the eigenvalues and to obtain higher-order eigenvalues, more grid points are required. This is not without disadvantages, however. Expanding the determinant becomes more difficult and finding the zeros of the higher-order polynomials is more difficult.

Table Solution of the Eigenproblem			
Z	k	$k(\text{exact})$	Error, %
1.618	± 3.090	$\pm \pi = \pm 3.142$	∓ 1.66
0.618	± 5.878	$\pm 2\pi = \pm 6.283$	∓ 6.45
-0.618	± 8.090	$\pm 3\pi = \pm 9.425$	∓ 14.16
-1.618	± 9.511	$\pm 4\pi = \pm 12.566$	∓ 24.31

Numerical Analysis

DWE3214

PART-III: Numerical Solutions of PDEs

Unit-8: Numerical Solutions of PDEs



Dr. Zaid Al-Azzawi

**University of Al-Anbar
College of Engineering**

2022/2023

Partial Differential Equations :

①

The Most commonly used PDE's in Engineering and Science are the second order P.D.E's, which have the following general Formulation:

$$A U_{xx} + B U_{xy} + C U_{yy} + D U_x + E U_y + F U + G = 0 \quad \text{--- ①}$$

Where A, B, C, D, E, F and G are functions of x and y , and the subscript refers to the partial derivatives:

like $U_{xy} = \frac{\partial^2 u}{\partial x \partial y}$

The general Formula (Eq. ①) is classified into three categories according to the following:

<u>Index</u>	<u>classification</u>
$B^2 - 4AC < 0$	Elliptic equation.
$B^2 - 4AC = 0$	Parabolic equation.
$B^2 - 4AC > 0$	Hyperbolic equation.

The partial differential Equations can also be classified physically into:

- 1- propagation problem.
- 2- Equilibrium problems.
- 3- Eigenvalue problems.

Note: The first order PDE's are allways hyperbolic.

(I) Elliptic Equation :

Poisson's Equation: $U_{xx} + U_{yy} = f(x, y)$

and Laplace Equation: $U_{xx} + U_{yy} = 0$

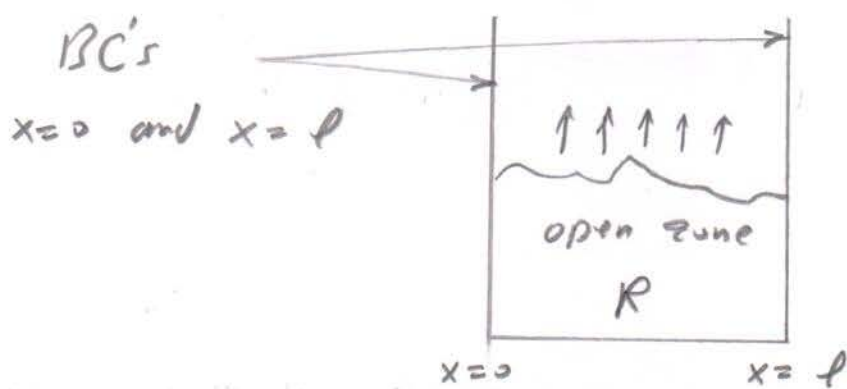
are both examples of Elliptic equation in two dimensions. The solution of these equation is the primitive function $U(x, y)$ which satisfies each point in the Range (R) enclosed by the boundary curve (C) .



(II) Parabolic Equation :

The one-dimensional conduction equation,

$U_t = K^2 U_{xx}$, is one of the most famous examples of parabolic equations. The solution to this equation is the primitive equation $U(x, t)$, which satisfies the spatial values of x between 0 and l and the time t from 0 to ∞ . The solution here is not bracketed in a closed domain, but it propagates in an open ended zone, and the solution needs one initial condition and two boundary conditions.



(III) Hyperbolic Equations:

The one dimensional wave equation,

$U_{tt} = c^2 U_{xx}$; is one of the first examples of hyperbolic equations.

The solution to this PDE's is the displacement primitive equation $u(x,t)$ for $x=0 \rightarrow l$ and $t=0 \rightarrow \infty$ that satisfies the two initial conditions and two boundary conditions. the solution also propagate in an open domain.

Finite Difference Approximation to partial Derivatives:

$$\frac{\partial u}{\partial x} = U_x = \frac{U_{i+1,j} - U_{i-1,j}}{2h}$$

$$h = \Delta x$$

$$K = \Delta y$$

$$\frac{\partial^2 u}{\partial x^2} = U_{xx} = \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2}$$

$$\frac{\partial u}{\partial y} = U_y = \frac{U_{i,j+1} - U_{i,j-1}}{2K}$$

$$\frac{\partial^2 u}{\partial y^2} = U_{yy} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{K^2}$$

Partial Differential Equations :

(3)

$$U_{xx} + U_{yy} = 0 \Rightarrow \text{Laplace Equation}$$

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{k^2} = 0$$

If we used a square grid where $h = k \Rightarrow$

$$U_{i,j} = \frac{1}{4} (U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1})$$

Where we can see that the value of $(U_{i,j})$ is the mean of the next 4-nodes; and it is called the (Standard Five points Equation). This equation can be written in another form:

$$U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j} = 0$$

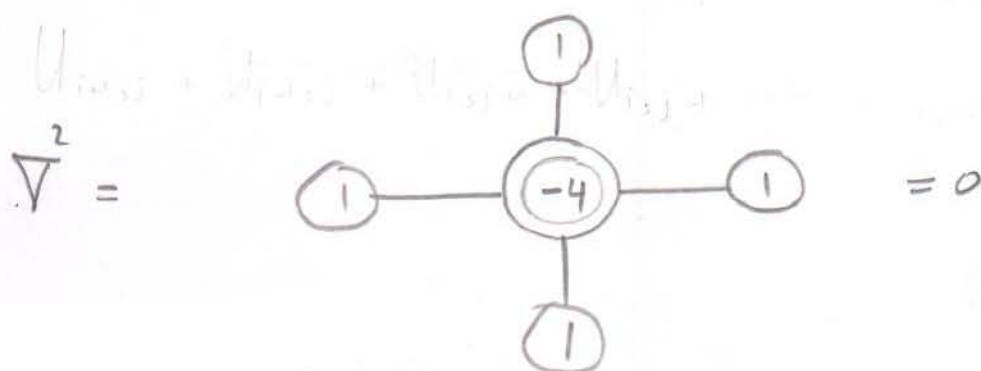
→ ① Elliptic Equations: (i) Laplace Equation:-

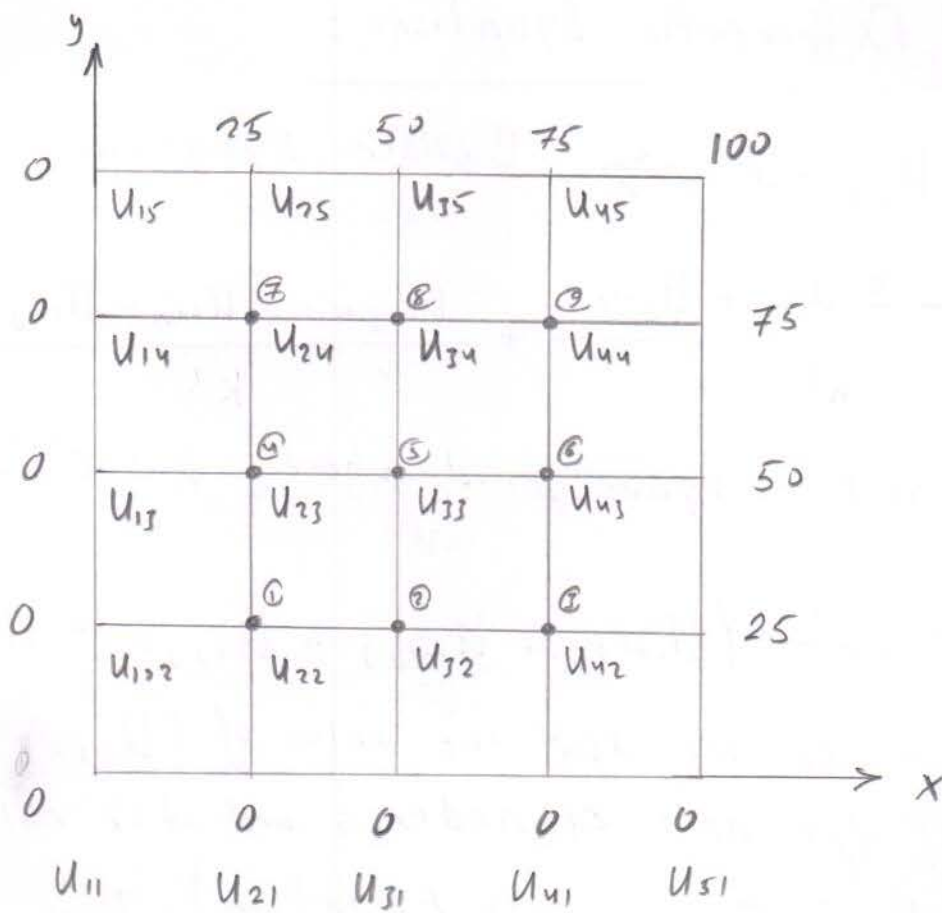
Ex: Solve $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ with the boundary conditions

$$U(0,y) = 0, \quad U(x,0) = 0, \quad U(x,l) = \underline{100x}, \quad U(l,y) = \underline{100y}$$

taking $h = 0.25l$?

Sol: Using Stencil form of FDE:





2,2

$$\cancel{U_{21}} + \cancel{U_{12}} + U_{23} + U_{32} - 4U_{22} = 0$$

3,2

$$\cancel{U_{31}} + U_{22} + U_{33} + U_{42} - 4U_{32} = 0$$

4,2

$$\cancel{U_{41}} + U_{32} + U_{43} + \cancel{U_{52}} - 4U_{42} = 0$$

2,3

$$U_{22} + \cancel{U_{13}} + U_{24} + U_{33} - 4U_{23} = 0$$

3,3

$$U_{32} + U_{23} + U_{34} + U_{43} - 4U_{33} = 0$$

4,3

$$U_{42} + U_{33} + U_{44} + \cancel{U_{52}} - 4U_{43} = 0$$

2,4

$$U_{23} + \cancel{U_{14}} + \cancel{U_{25}} + U_{34} - 4U_{24} = 0$$

3,4

$$U_{33} + U_{24} + \cancel{U_{35}} + U_{44} - 4U_{34} = 0$$

4,4

$$U_{43} + U_{34} + \cancel{U_{45}} + \cancel{U_{54}} - 4U_{44} = 0$$

$$-4 U_{22} + U_{32} + U_{23} + U_{23}$$

$$= 0$$

$$\textcircled{4} = 0$$

$$U_{22} - 4 U_{32} + U_{42} + U_{33} + U_{33}$$

$$= 0$$

$$= 0$$

$$U_{32} - 4 U_{42} + U_{43}$$

$$= -25$$

$$U_{22} - 4 U_{23} + U_{33} + U_{24}$$

$$= 0$$

$$U_{32} + U_{23} - 4 U_{33} + U_{43} + U_{34}$$

$$= 0$$

$$U_{42} + U_{33} - 4 U_{43} + U_{44} = -50$$

$$+ U_{23} - 4 U_{24} + U_{34} = -25$$

$$+ U_{33}$$

$$+ U_{24} - 4 U_{34} + U_{44} = -50$$

$$U_{43}$$

$$+ U_{34} - 4 U_{44} = -150$$

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} U_{22} \\ U_{32} \\ U_{42} \\ U_{24} \\ U_{33} \\ U_{43} \\ U_{24} \\ U_{34} \\ U_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -25 \\ 0 \\ 0 \\ -50 \\ -25 \\ -50 \\ -150 \end{bmatrix}$$

\Rightarrow

$$U_{22} = 6.25 \quad U_{32} = 12.5 \quad U_{42} = 18.75$$

$$U_{24} = 12.5 \quad U_{33} = 25 \quad U_{43} = 37.5$$

$$U_{24} = 18.75 \quad U_{34} = 37.5 \quad U_{44} = 56.25$$

Ex Solve the Poisson's equation

H.W

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -36(x^3 + y^3 + 5)$$

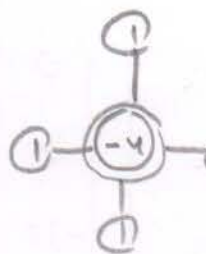
subject to the conditions:

$$u = 0 \text{ at } x = 0 \text{ and } x = 1$$

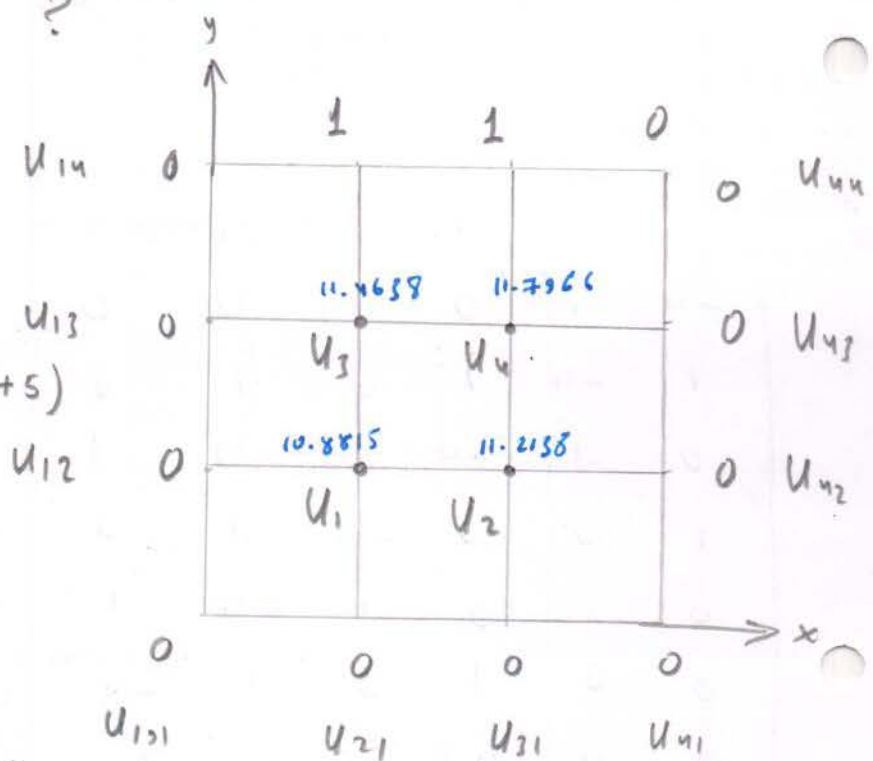
$$u = 0 \text{ at } y = 0$$

$$u = 1 \text{ at } y = 1 \text{ in } 0 < x < 1$$

use $(h = \frac{1}{3}l)$?



$$= -36 \left(\frac{1}{3}l\right)^2 (x_i^3 + y_j^3 + 5)$$



Sub. to get 4-eg's.

and solve ?

يمكن تبسيط x_i, y_j (*)

من طريقة استقرارية i, j

ملاحظة u_1, u_2, u_3, u_4

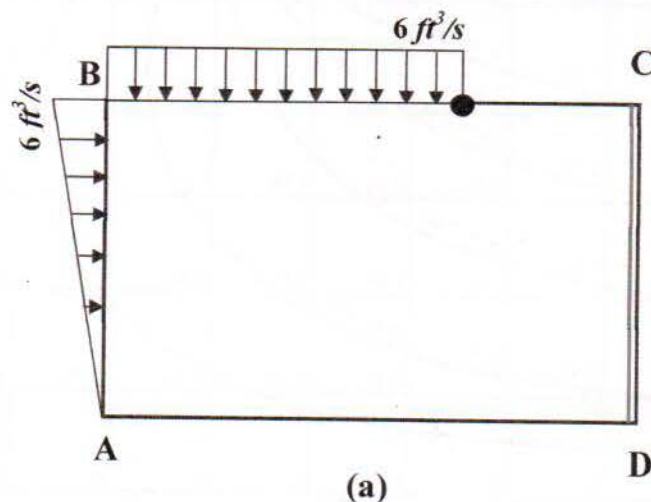
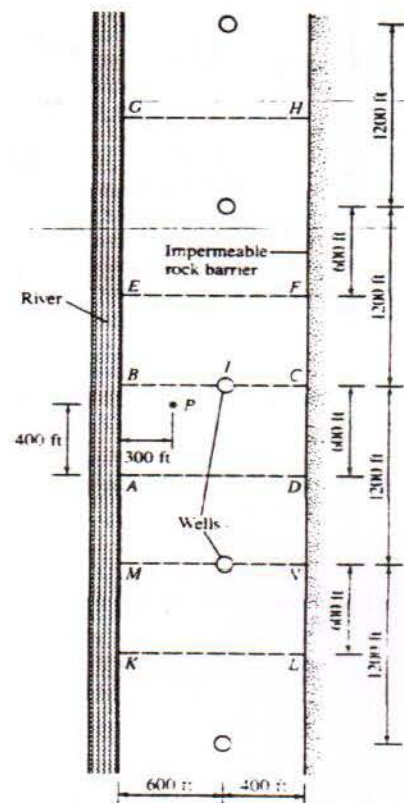
$i=2, j=2$ $i=1, j=2$ $i=2, j=1$ $i=1, j=1$

هذا يعني $h = \frac{1}{3}$ نقطة

Example: A line of wells to be drilled in an alluvial aquifer as shown in the next Figure. The wells are equally spaced along a line 600 ft from the river. An impermeable barrier is located parallel to the river and 1000 ft from the river. Each well is pumped at a rate of $12 \text{ ft}^3/\text{s}$. Determine the streamline and constant piezometric head pattern, and the velocity at point P. Assume that the aquifer is confined with a depth of 100 ft.

Solution:

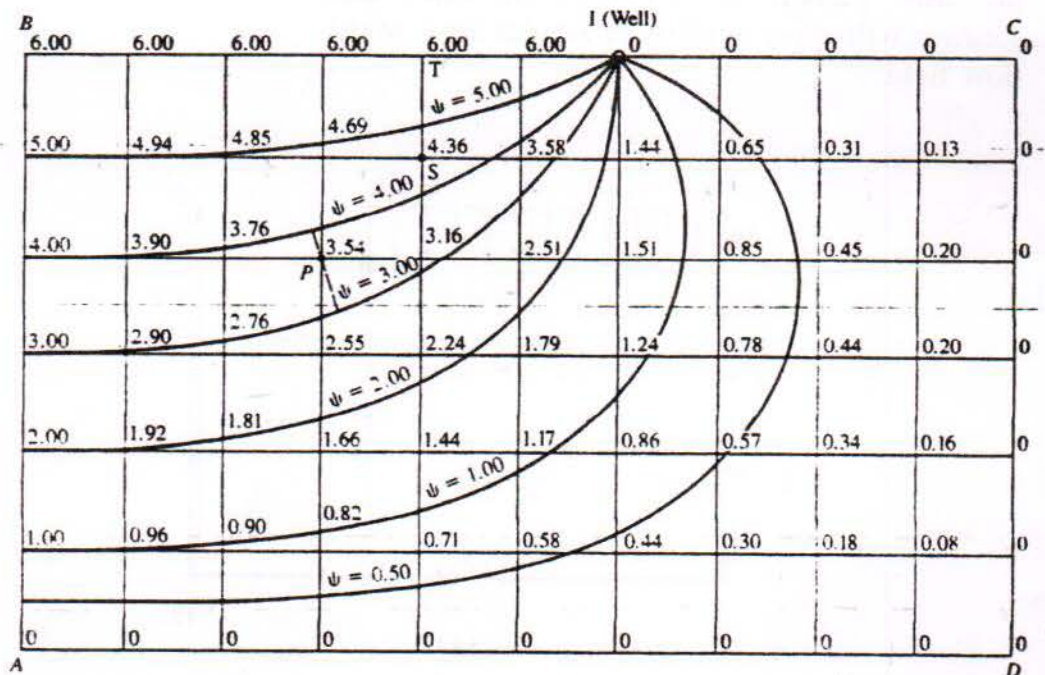
Considering the geometry of the flow field will show us that the flow pattern in AEFD will have a flow pattern exactly like that within EGHF. Thus, we need to solve for the flow pattern within AEFD only, since that solution will apply to EGHF and other similar areas such as ADKL. Further consideration of the geometry of AEFD also shows that we can expect the flow pattern within ABCD to be mirror image of that within BEFC or ADMN. Therefore we need to solve for the flow pattern in ABCD only, since that solution will apply to all other similar areas in the flow field.



$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0$$

Using the same Technique and $h = 100$ ft
to define 45 interior node and 32 Boundary
node. We get

$$\nabla^2 \psi = 0$$



(II) Parabolic Equations :

6

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Assuming a rectangular grid in the $(x-t)$ plane ; such as

$$x = i h \quad (i = 0, 1, 2, 3, \dots)$$

$$t = j K \quad (j = 0, 1, 2, 3, \dots) \quad \Rightarrow$$

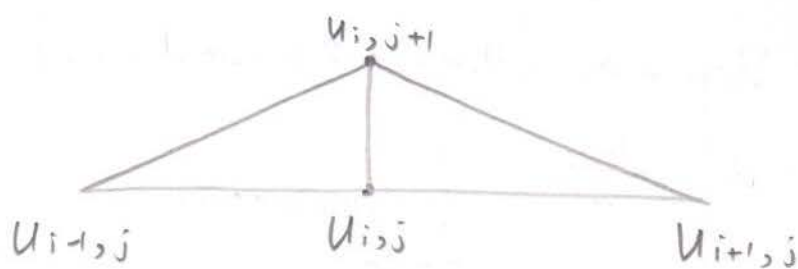
$$\frac{1}{K} (U_{i,j+1} - U_{i,j}) = \frac{1}{h^2} (U_{i+1,j} - 2U_{i,j} + U_{i-1,j})$$

Forward Diff.

$$\therefore U_{i,j+1} = \alpha (U_{i+1,j} + U_{i-1,j}) + (1 - 2\alpha) U_{i,j}$$

$$\text{where } \alpha = \frac{K}{h^2}$$

and this is called the Explicit scheme.



The solution of the Explicit scheme is stable when $\alpha \leq \frac{1}{2}$; and when $\alpha = \frac{1}{2}$ the equation becomes :

$$U_{i,j+1} = \frac{1}{2} (U_{i+1,j} + U_{i-1,j})$$

Ex] Obtain the Numerical Solution of the differential equation :

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad , \quad 0 < x < 1, t \geq 0$$

under the condition that :

$$U(0, t) = U(1, t) = 0.0 \text{ and}$$

$$U(x, 0) = 2x \quad \text{for} \quad 0 \leq x \leq \frac{1}{2}$$

$$= 2(1-x) \quad \text{for} \quad \frac{1}{2} \leq x \leq 1$$

solve using the explicit scheme ?

Sol $\frac{1}{K} (U_{i,j+1} - U_{i,j}) = \frac{1}{h^2} (U_{i-1,j} - 2U_{i,j} + U_{i+1,j})$

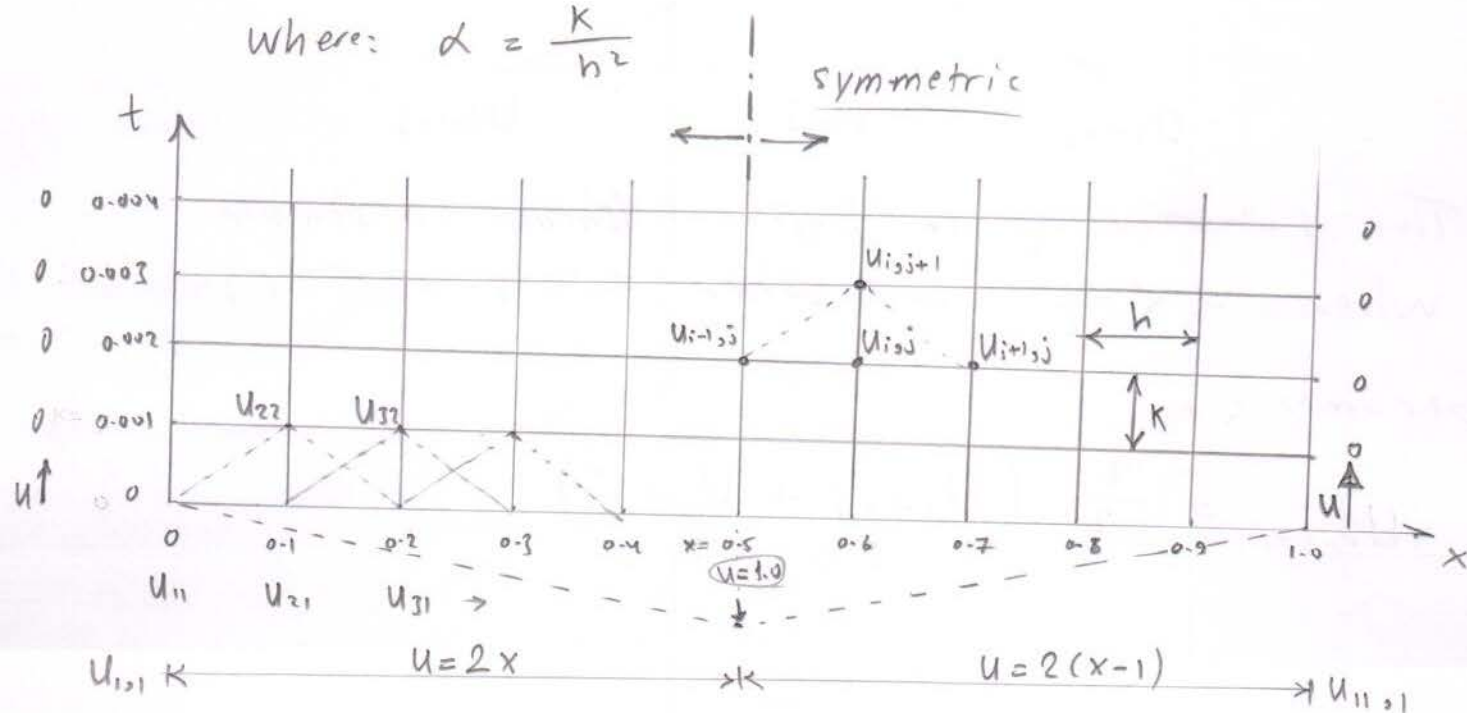
Where : $x = i \cdot h \quad (i = 0, 1, 2, \dots)$

$t = j \cdot K \quad (j = 0, 1, 2, \dots)$ and

$$U_{i,j+1} = U_{i,j} + \frac{K}{h^2} (U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) \quad \text{or}$$

$$U_{i,j+1} = U_{i,j} + \alpha (U_{i-1,j} - 2U_{i,j} + U_{i+1,j})$$

Where : $\alpha = \frac{K}{h^2}$



Sol₂

$$\alpha = \frac{k^2}{h^2} = 1.0 \Rightarrow$$

$$U_{i,j+1} = U_{i-1,j} + U_{i+1,j} - U_{i,j-1} \quad (*)$$

B.C's and I.C's:

$$U_{0,j} = U_{1,j} = 0 \quad (1)$$

Initial conditions:

$$U_{i,0} = \frac{1}{2} X_i (1 - X_i) \quad (2)$$

$$\frac{\partial u}{\partial t} = 0 \Rightarrow \frac{U_{i,j+1} - U_{i,j-1}}{2k} = 0 \quad \text{at } t=0$$

$$\Rightarrow U_{i,1} - U_{i,-1} = 0 \Rightarrow$$

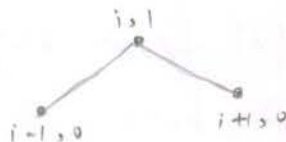
$$U_{i,1} = U_{i,-1} \quad (3)$$

Because of the symmetry we will solve only for

$$i=1 \rightarrow i=5. \Leftrightarrow x=0.1 \rightarrow x=0.5$$

$$\text{For } \underline{j=0} \Rightarrow U_{i,1} = U_{i,-1} \quad (3) \text{ sub. in } (*) \Rightarrow$$

$$U_{i,1} = \frac{1}{2} (U_{i-1,0} + U_{i+1,0})$$



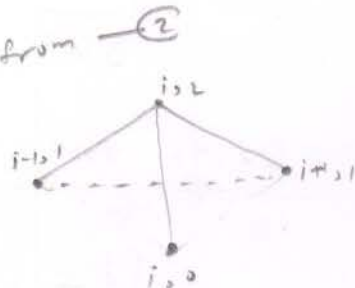
For $j=1, 2, 3, 4, 5$ and From $(*)$

$$j=1 \quad U_{i,2} = U_{i-1,1} + U_{i+1,1} - U_{i,0} \quad \text{from } (2)$$

$$j=2 \quad U_{i,3} = U_{i-1,2} + U_{i+1,2} - U_{i,1}$$

$$j=3 \quad U_{i,4} = U_{i-1,3} + U_{i+1,3} - U_{i,2}$$

$$j=4 \quad U_{i,5} = U_{i-1,4} + U_{i+1,4} - U_{i,3}$$



$$i=1 \rightarrow 5$$

Let $h = 0.1$; $K = 0.001 \Rightarrow \alpha = \frac{K}{h^2} = 0.1$

(7)

$$U_{i,j+1} = U_{i,j} + \alpha (U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) \Rightarrow$$

$$U_{i,j+1} = \frac{1}{10} U_{i,j} + \frac{1}{10} (U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) \Rightarrow$$

$$U_{i,j+1} = \frac{1}{10} (U_{i-1,j} + 8U_{i,j} + U_{i+1,j}) \quad \text{--- (*)}$$

Substitute in each Unknown points :

For $j=1 \Rightarrow$ to find $j+1$

$$i=2 \Rightarrow U_{2,2} = \frac{1}{10} (U_{1,1} + 8U_{2,1} + U_{3,1}) = \frac{1}{10} (0 + 8(0.2) + 0.4) = 0.2$$

$$i=3 \Rightarrow U_{3,2} = \frac{1}{10} (U_{2,1} + 8U_{3,1} + U_{4,1}) = \frac{1}{10} (0.2 + 8(0.4) + 0.6) = 0.4$$

$$i=4 \Rightarrow U_{4,2} = \frac{1}{10} (U_{3,1} + 8U_{4,1} + U_{5,1}) = \frac{1}{10} (0.4 + 8(0.6) + 0.8) = 0.6$$

$$i=5 \Rightarrow U_{5,2} = \frac{1}{10} (U_{4,1} + 8U_{5,1} + U_{6,1}) = \frac{1}{10} (0.6 + 8(0.8) + 1.0) = 0.8$$

$$i=6 \Rightarrow U_{6,2} = \frac{1}{10} (U_{5,1} + 8U_{6,1} + U_{7,1}) = \frac{1}{10} (0.8 + 8(1.0) + 0.8) = 0.96$$

and From Symmetry :-

$$U_{7,2} = U_{5,2} = 0.8$$

$$U_{8,2} = U_{4,2} = 0.6$$

$$U_{9,2} = U_{3,2} = 0.4$$

$$U_{10,2} = U_{2,2} = 0.2$$

$$U_{11,2} = U_{1,2} = 0.0$$

and by same method using $j=2$ & $i=2 \rightarrow 11$

$j=3$ & $i=2 \rightarrow 11$

$j=n$ & $i=2 \rightarrow 11$

	$i=0$	1	2	3	4	5	6
$x=0$	0.0	0.1	0.2	0.3	0.4	0.5	0.6
$j=0, t=0.000$	0.0	0.200	0.40	0.60	0.80	1.00	0.8
$j=1, t=0.001$	0.0	0.2	0.40	0.60	0.80	0.960	
$j=2, t=0.002$	0.0	0.2	0.40	0.60	0.7960	0.928	
$j=3, t=0.003$	0.0	0.2	0.40	0.5996	0.7896	0.9016	Symm.
$j=4, t=0.004$	0.0	0.2	0.40	0.5986	0.7818	0.8792	
$j=5, t=0.005$	0.0	0.2	0.3999	0.5971	0.7732	0.8597	
$j=10, t=0.01$	0.0	0.1996	0.3968	0.5822	0.7281	0.7867	
$j=20, t=0.02$	0.0	0.1938	0.3781	0.5373	0.6486	0.6891	

H.W. Consolidation Example

(III)

Hyperbolic Equation :

(8)

(i) Wave Equation: $\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} \quad 0 \leq x \leq l$
 $t \geq 0$

$$\frac{1}{K^2} (U_{i,j-1} - 2U_{i,j} + U_{i,j+1}) = \frac{1}{h^2} (U_{i-1,j} - 2U_{i,j} + U_{i+1,j})$$

where $x = i \cdot h \quad (i = 0, 1, 2, \dots)$

$t = j \cdot K \quad (j = 0, 1, 2, \dots)$

Let $\alpha = \frac{K}{h} \Rightarrow$

$$U_{i,j+1} = \alpha^2 (U_{i-1,j} + U_{i+1,j}) + 2(1 - \alpha^2) U_{i,j} - U_{i,j-1}$$

This equation is conditionally stable at $\alpha \leq 1.0$

Ex | The transverse displacement (u) of a point at a distance (x) from any end at any time (t) of a vibrating string satisfies the equation :

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2} \quad 0 \leq x \leq 1, \quad t \geq 0$$

With the B.C's :

$$U = 0 \quad \text{at} \quad x = 0, \quad t \geq 0$$

$$U = 0 \quad \text{at} \quad x = 1, \quad t \geq 0$$

with the I.C's :

$$U = \frac{1}{2}x(1-x) \quad \text{and} \quad \frac{\partial u}{\partial t} = 0 \quad \text{at} \quad t = 0, \quad 0 \leq x \leq 1$$

Solve the above equation for $(0 \leq x \leq 1.0, 0 \leq t \leq 0.4)$ with $h = K = 0.1$?

Initially the values will be as follows:

⑨

initially

← \uparrow \rightarrow symm.

X	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$j=0$ $u = \frac{1}{2}x(1-x)$	0	0.045	0.08	0.105	0.12	0.1	0.12	0.105	0.08	0.045	0

For other j 's use the previous equations $j=0 \rightarrow j=4$

