German University in Cairo
الجامعة الألمانية بالقاهرة

Numerical Analysis

(ENME 602)

Spring 2021

Lecture 1

Dr. Hesham H. Ibrahim
Associate Professor, Mechatronics Department.
hesham.hamed-ahmed@guc.edu.eg
Office C7.04

Course Description

Course Objectives

- Recognize numerical solution methods for different types of problems (mathematical models).
- Categorize numbers and errors occurring on computers.
- Find roots of Nonlinear equations by different Numerical Techniques.
- Approximate functions and interpolate function values.
- Solve linear systems of equations.
- Integrate functions numerically using different effective methods.
- Solve ordinary differential equations by different methods.

Course Content

- ✓ Introduction
- ✓ Round-off Errors and Computer Arithmetic
- ✓ Algorithms and Convergence
- ✓ Roots of Non-linear Equations
- ✓ Interpolation and Polynomial Approximation
- ✓ Linear Systems: Iterative Methods
- ✓ Numerical Integration
- ✓ Numerical Solution of Differential Equations
- ✓ Minimization Problems (Least Squares Approximation)

Course Assessment

Home and Class Work	**10 %**
3 Quizzes (best 2)	**20 %**
Mid Term Exam	**30 %**
Final Exam	**40 %**

Fundamental Course Agreement

- No late submission is accepted
- No excuses are accepted for missing a quiz
- Best two out of three quizzes are counted.

Recommended Textbooks

- Richard L. Burden, J. Douglas Faires, and Annette Burden, Numerical Analysis, Brooks/Cole CENGAGE Learning, 9th Edition. ISBN: 978-0-538-73564-3.
- G. Baumann (2010). Mathematics for Engineers IV: Numerics. Munich: Oldenbourg. ISBN: 978-3-486-59042-5.
- S.C. Chapra and R.P. Canale (2003). Numerical Methods for Engineers. Boston: McGraw Hill. ISBN: 0073101567.

Acknowledgment

We would like to thank Professor Annette Burden, one of the authors of the text book below*, for granting us the permission to use their online materials at https://sites.google.com/site/numericalanalysis1burden/05-power-points/dr-b-s-help-files

*Richard L. Burden, J. Douglas Faires, Annette Burden, Numerical Analysis, Brooks/Cole CENGAGE Learning, 9th Edition. ISBN: 978-0-538-73564-3.

Preparing for the Course & Self-Study

The prerequisites: Review the analytical methods for solving the same set of problems. The exact solution, whenever available, is needed often to test the accuracy of the approximations.

Computer package & Calculators: Algorithms are run using Matlab during tutorials for demonstration and non-programmable calculators are mandatory and used in tutorials and exams.

Applications: Practice sheets contains concept checking questions as well as engineering context problems.

Lecture 1

"Introduction and Computer Arithmetic"

Lecture 1

Introduction and Computer Arithmetic

1.1 Introduction

1.2 Main Types of Problems to be Solved Numerically

1.3 Round-off Errors and Computer Arithmetic

1.1 Introduction

1.1 Introduction

- *Numerical Analysis* is the study of algorithms that use numerical approximation (as opposed to symbolic manipulations) for the problems of mathematical analysis.
- *Numerical Analysis* naturally finds application in all fields of engineering and the physical sciences, but in the 21st century also the life sciences, social sciences, medicine, business and even the arts have adopted elements of scientific computations.
- The growth in computing power has revolutionized the use of realistic mathematical models in science and engineering, and subtle numerical analysis is required to implement these detailed models of the world.
- For example, ordinary differential equations appear in celestial mechanics (predicting the motions of planets, stars and galaxies); numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology.

1.1 Introduction

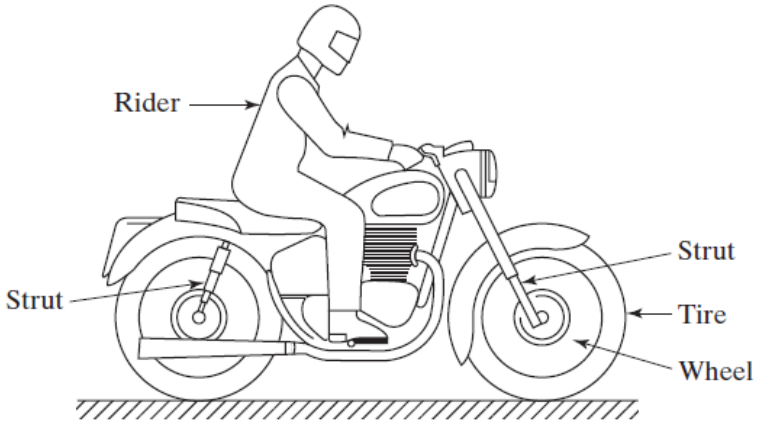
- Virtually every phenomenon in nature, whether biological, geological or mechanical, can be described with the aid of laws of physics in terms of algebraic, differential, or integral equations relating various quantities of interest.

- Development of the mathematical model of a process is achieved through assumptions concerning how the process works (ex. Continuous contact, neglect inertia effect, temperature independent properties, etc.)

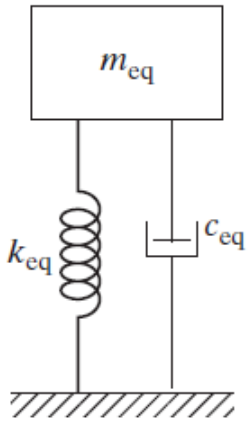
- Most engineers and scientists studying physical phenomena are involved with two major tasks:
  1. Mathematical formulation of the physical process. (Modelling)
  2. Numerical analysis of the mathematical model.

1.1 Introduction

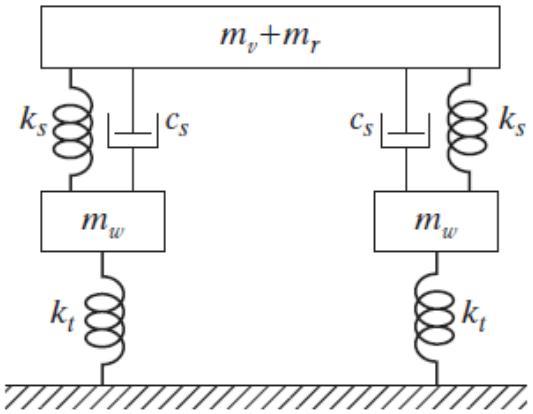
- Modelling and Simulation
  - ✓ **Step 1: Mathematical Modelling**



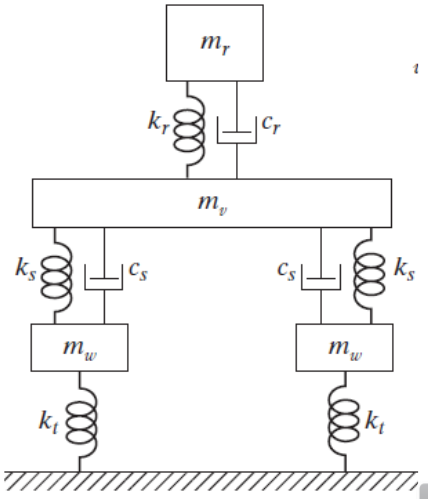
(a)



(b)



(c)



(d)

1.1 Introduction

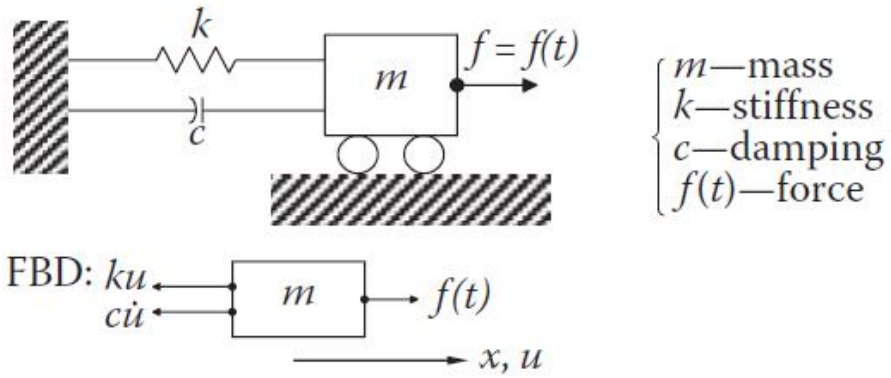
- Modelling and Simulation

  ✓ **Step 2: Derivation of Governing Equations**

  - Once the mathematical model is available, we use the principles of dynamics and derive the equations that describe the vibration of the system.
  - The equations of motion can be derived conveniently by drawing the free-body diagrams of all the masses involved. The free-body diagram of a mass can be obtained by isolating the mass and indicating all externally applied forces, the reactive forces, and the inertia forces.
  - The equations of motion of a vibrating system are usually in the form of a set of ordinary differential equations for a discrete system and partial differential equations for a continuous system.
  - The equations may be linear or nonlinear, depending on the behavior of the components of the system. Several approaches are commonly used to derive the governing equations. Among them are Newton's second law of motion, D'Alembert's principle, and the principle of conservation of energy.

1.1 Introduction

- Modelling and Simulation
  - ✓ **Step 2: Derivation of Governing Equations**



$$m\ddot{u} = f(t) - k\,u - c\,\dot{u}$$

$$m\,\ddot{u} + c\,\dot{u} + k\,u = f(t)$$

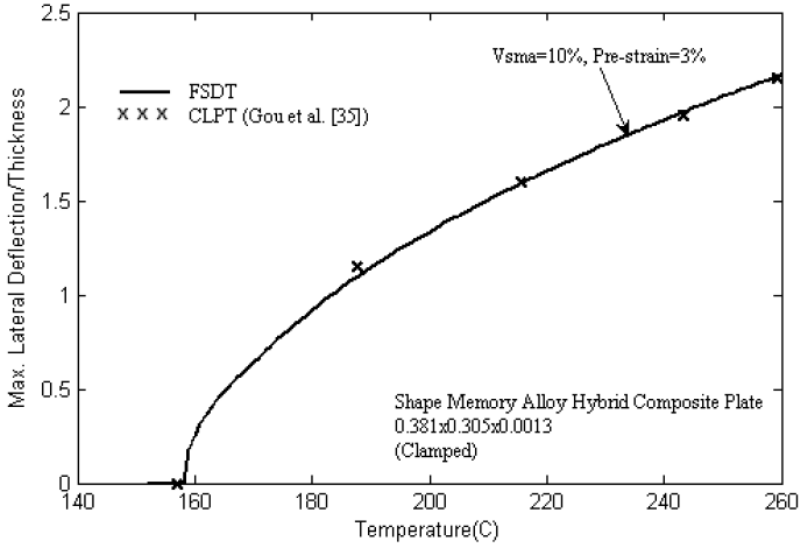
- Modal analysis ($f(t) = 0$)
- Frequency response analysis ($f(t) = F \sin \omega t$)
- Transient response analysis ($f(t)$ is arbitrary)

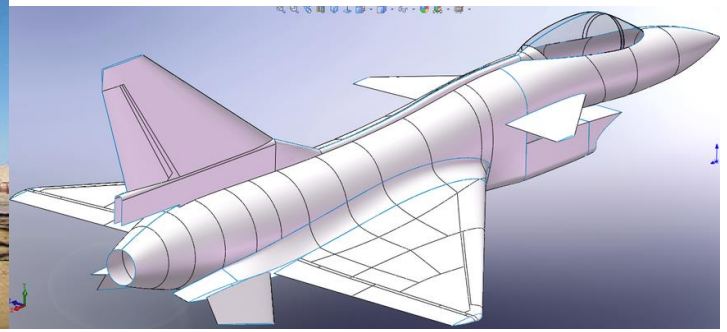
1.1 Introduction

- Modelling and Simulation

  ✓ **Step 2: Derivation of Governing Equations**

$$[M]\{\ddot{W}\}+[G]\{\dot{W}\}+\left(\lambda[A_a]+[K]-[K_T]+[K_r]+\frac{1}{2}[N1]+\frac{1}{3}[N2]\right)\{W\}$$
$$=\{P_T\}-\{P_r\}$$





1.1 Introduction

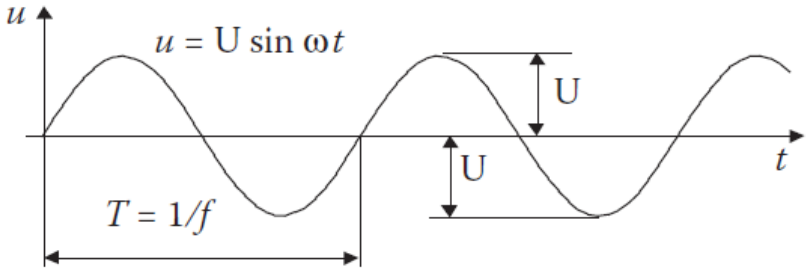
- Modelling and Simulation
  - ✓ **Step 3: Solution of the Governing Equations**
    - The equations of motion must be solved to find the response of the vibrating system.
    - Depending on the nature of the problem, we can use one of the following techniques for finding the solution: standard methods of solving differential equations, Laplace transform methods, and numerical methods.
    - If the governing equations are nonlinear, they can seldom be solved in closed form. Furthermore, the solution of partial differential equations is far more involved than that of ordinary differential equations.
    - Numerical methods involving computers can be used to solve the equations.

1.1 Introduction

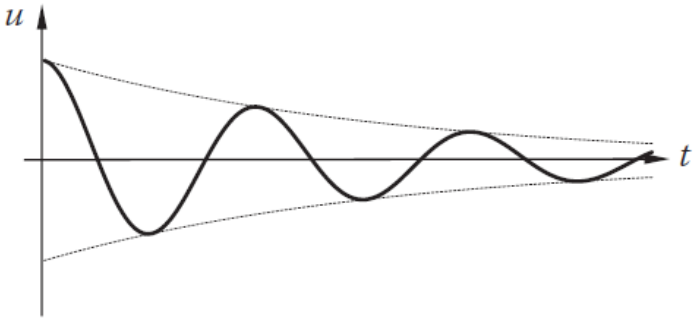
- Modelling and Simulation
  - ✓ **Step 4: Interpretation of the Results**

    - The solution of the governing equations could give displacements, velocities, and accelerations of the various masses of the system.
    - These results must be interpreted with a clear view of the purpose of the analysis and the possible design implications of the results.

Undamped Free Vibration



Damped Free Vibration



1.2 Main Types of Problems to be Solved Numerically

1.2 Main Types of Problems to be Solved Numerically

- Non-linear equations in one variable
- Systems of Linear Equations, using iterative techniques
- Data fitting and interpolation of functions
- Numerical approximation of derivatives and integrals
- Numerical solutions of ordinary differential equations (Initial value problems & Boundary value problems)

For each problem solving, the engineer's mission, is to study the algorithm's efficiency and convergence:

- Error analysis & Convergence: Computer numbers and types of errors (Round-off errors, relative and absolute errors, error bounds)

1.2 Main Types of Problems to be Solved Numerically

- Non-linear equations in one variable

A sample problem: Solve the non-linear equation for the variable h

$$12.4 = 10\left[0.5\pi - \sin^{-1}(h) - h\sqrt{1-h^2}\right]$$

The analytical solution (or exact solution) doesn't exist.

The numeric approximation of the solution, using Matlab is

$$h = 0.16617$$

1.2 Main Types of Problems to be Solved Numerically

- Non-linear equations in one variable

The root finding problem: Approximate the solution $x \in \mathbb{R}$, root of the non-linear equation:

$$f(x) = 0$$

The iterative techniques used are:

- Bisection Method
- Newton's method
- Variations of Newton's method: Secant method, False position method
- Fixed point method

1.2 Main Types of Problems to be Solved Numerically

- Systems of Linear Equations, using Iterative Techniques

A sample problem: Solve the following linear system:

$$\begin{cases} 10x_1 - x_2 + 2x_3 = 6 \\ -x_1 + 11x_2 - x_3 + x_4 = 25 \\ 2x_1 - x_2 + 10x_3 - 3x_4 = -11 \\ 3x_2 - x_3 + 8x_4 = 15 \end{cases}$$

The unique solution using Linear Algebra techniques is:

$$\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4) = (1, 2, -1, 1)^t$$

The numerical solution is an approximation $\mathbf{x^{(k)}}$ of the vector $\bar{\mathbf{x}}$, obtained using iterative techniques: $\mathbf{x^{(5)}} \simeq \bar{\mathbf{x}}$

$\mathbf{x^{(k)}}$	$\mathbf{x^{(0)}}$	$\mathbf{x^{(1)}}$	$\mathbf{x^{(2)}}$	$\mathbf{x^{(3)}}$	$\mathbf{x^{(4)}}$	$\mathbf{x^{(5)}}$
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	-0.9873	-1.014	-1.0025	-1.0003	-1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

1.2 Main Types of Problems to be Solved Numerically

- Systems of Linear Equations, using Iterative Techniques

The general problem: Approximate the vector solution $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n)^t \in \mathbb{R}^n$ of the following linear system:

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

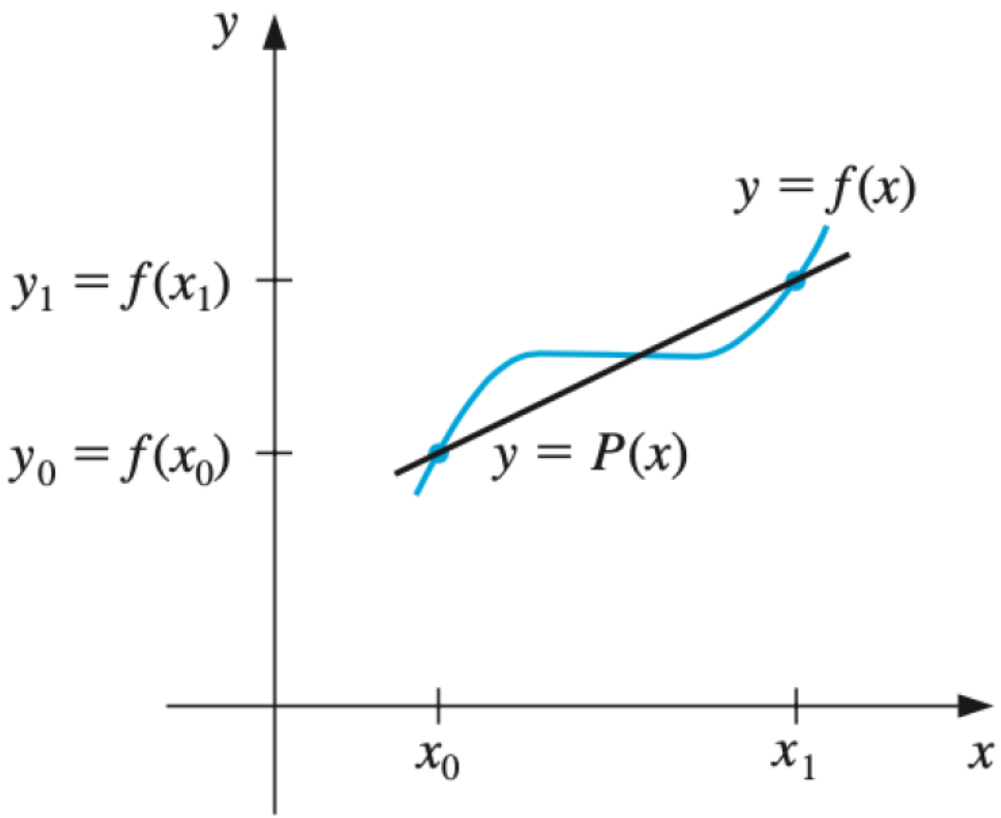
The numerical solution is an approximation $\tilde{\mathbf{x}} \in \mathbb{R}^n$ of the vector $\bar{\mathbf{x}}$, $(\tilde{\mathbf{x}} \simeq \bar{\mathbf{x}})$ obtained using the following iterative techniques:

- Jacobi method
- Gauss-Seidel method
- Successive Over-relaxation method (SOR)

1.2 Main Types of Problems to be Solved Numerically

- Data fitting and interpolation of functions

Linear interpolation problem: Construct a linear function $P(x)$ that passes through two nodes $(x_0, f(x_0))$, $(x_1, f(x_1))$

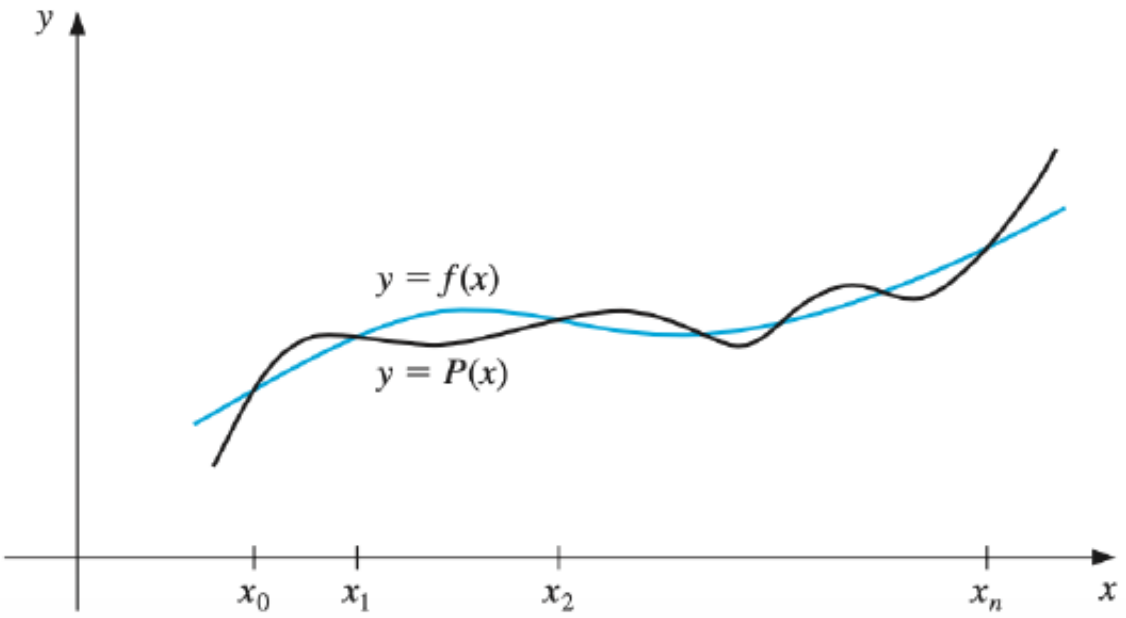


1.2 Main Types of Problems to be Solved Numerically

- Data fitting and interpolation of functions

Non-linear interpolation problem: Construct a polynomial $P(x)$ of degree at most n that passes through $(n+1)$ nodes

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



1.2 Main Types of Problems to be Solved Numerically

- Data fitting and interpolation of functions

The general problem: Use an algebraic polynomial to approximate an arbitrary set of data

Linear interpolation problem: Construct a linear function $P(x)$ that passes through two nodes (x_0, y_0), (x_1, y_1)

General interpolation problem: Construct a polynomial $P(x)$ of degree at most n that approximates the function $f(x)$ and passes through $(n+1)$ nodes

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

The interpolation polynomials are

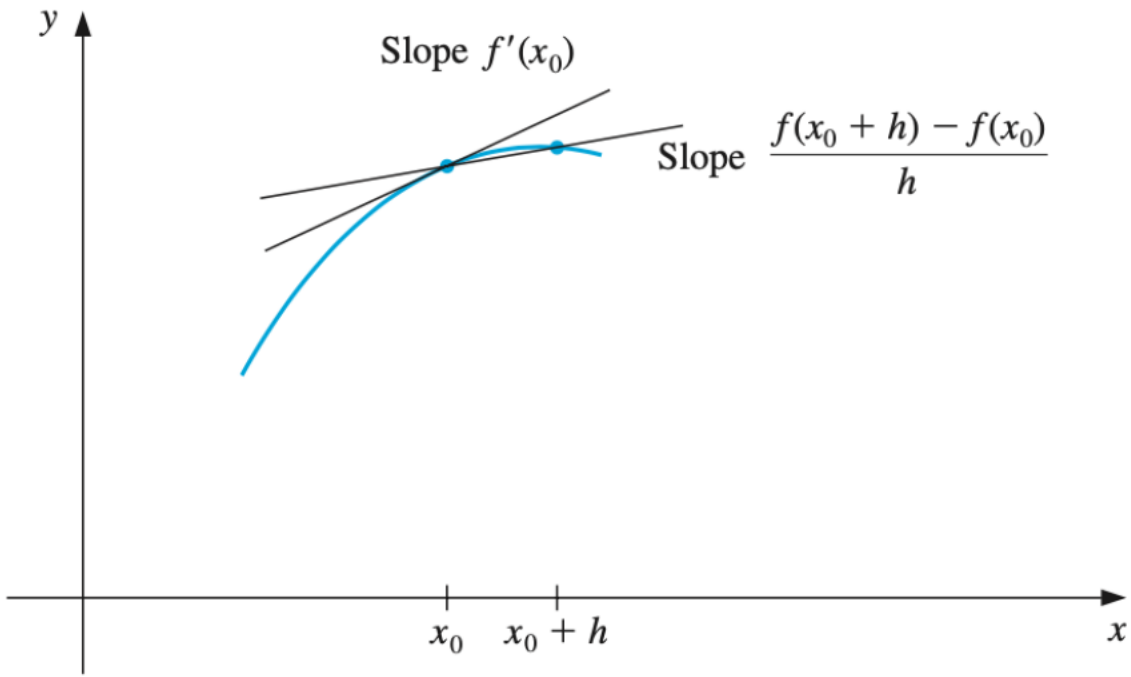
- Lagrange polynomials
- Newton's Finite Difference polynomials

1.2 Main Types of Problems to be Solved Numerically

- Numerical approximation of derivatives and integrals

Numerical Differentiation

Linear case: Approximate the slope of a function $f(x)$ using two nodes



1.2 Main Types of Problems to be Solved Numerically

- Numerical approximation of derivatives and integrals

Numerical Differentiation

General case: Approximate the slope of a function $f(x)$ at $(n+1)$ nodes

Example: Approximate $f'(x)$ at 6 nodes, given $f(x)$

x	$f(x)$	$f'(x)$
2.1	−1.709847	
2.2	−1.373823	
2.3	−1.119214	
2.4	−0.9160143	
2.5	−0.7470223	
2.6	−0.6015966	

The main numerical formulas used are

- Forward-Difference (Backward-Difference) formulas
- 3-points formulas
- 5-points formulas

1.2 Main Types of Problems to be Solved Numerically

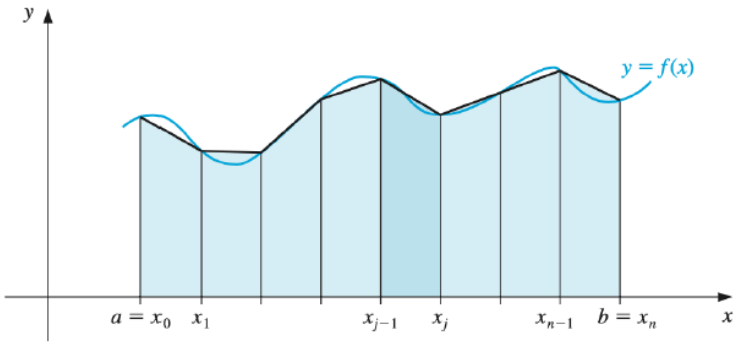
- Numerical approximation of derivatives and integrals

Numerical Integration

The general problem: Approximate the integral

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

Geometrically: Approximate the area under the curve $f(x)$ over the interval $[a, b]$



The main quadrature formulas (simple & composite) used are

- Trapezoidal rule
- Mid-point rule
- Simpson's rule

1.2 Main Types of Problems to be Solved Numerically

- Iterative techniques for solving initial value problems

The general problem: Approximate the solution function $\mathbf{y}(\mathbf{x}) \in \mathbb{R}$ of the following initial value problem:

$$\begin{cases} \dfrac{dy}{dx} &= f(x,y), \quad x \in [a,b] \\ y(a) &= y_0 \end{cases}$$

The numerical solution is an approximation $\tilde{\mathbf{y}}(\mathbf{x})$ of the solution function $\mathbf{y}(\mathbf{x})$, $\tilde{\mathbf{y}}(\mathbf{x}) \simeq \mathbf{y}(\mathbf{x})$ that fits the solution $\tilde{\mathbf{y}}(\mathbf{x_i}) = \mathbf{y}(\mathbf{x_i})$ at the approximation nodes $\mathbf{x_i}$, $i = 0, 1, 2, \ldots, n$.

The numerical methods used are

- Euler's method
- Taylors' method
- Runge-Kutta method of second and fourth order

1.2 Main Types of Problems to be Solved Numerically

- Iterative techniques for solving boundary value problems

The general problem: Approximate the solution function $\mathbf{y(x)} \in \mathbb{R}$ of the following 2^{nd} order boundary value problem:

$$\begin{cases} \dfrac{d^2y}{dx^2} + p(x)\dfrac{dy}{dx} + q(x)y = f(x), \quad x \in [a,b] \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

The numerical solution is an approximation $\tilde{\mathbf{y}}(\mathbf{x})$ of the solution function $\mathbf{y(x)}$, $\tilde{\mathbf{y}}(\mathbf{x}) \simeq \mathbf{y(x)}$ that fits the solution $\tilde{\mathbf{y}}(\mathbf{x_i}) = \mathbf{y(x_i)}$ at the approximation nodes $\mathbf{x_i}$, $\quad i = 0, 1, 2, \ldots, n.$

The numerical method used is

- The Finite Difference Method

1.3 Round-off Errors and Computer Arithmetic

1.3 Round-off Errors and Computer Arithmetic

- Mathematics and Computer arithmetics
- Floating-point form
- Arithmetic using k-digit rounding and chopping
- Floating-point form (general definition for or any base β)
- Floating-point form (underflow and overflow)
- Types of errors
- Errors as measures of accuracy
- Round-off errors

1.3 Round-off Errors and Computer Arithmetic

Mathematics versus Computer arithmetics

The arithmetic performed by a calculator or computer is different from the arithmetic in algebra and calculus courses.

Mathematical arithmetics (in theory) give exact values and allow infinite decimal expansions (expressed using "..."):

$$(\sqrt{3})^2 = 3, \quad \frac{2}{7} = 0.28571\ldots, \quad \frac{1}{3} = 0.3333\ldots$$

Computer arithmetic can only use a finite number of digits to represent numbers: check with MATLAB®.

1.3 Round-off Errors and Computer Arithmetic

Mathematics versus Computer arithmetics

MATLAB® uses either 5 digits or 16 digits format to display numbers.

Command Window

```
>> format short
>> pi
ans =
    3.1416
>> format long
>> pi
ans =
   3.141592653589793
fx >>
```

By default, MATLAB® uses 16 digits of precision, in computations regardless of display format. For higher precision, the *vpa* (Variable precision arithmetic) function in Symbolic Math Toolbox™, uses 32 significant decimal digits of precision.

1.3 Round-off Errors and Computer Arithmetic

Floating-point form

Normalized decimal floating-point form: Machine numbers are represented in the normalized decimal floating-point form

$$\pm 0.d_1 d_2 \dots d_{k-1} d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad 0 \le d_i \le 9,$$

for each $i = 2, \dots k$.
Numbers of this form are called *k-digit decimal machine numbers*.

- π has an infinite decimal expansion

$$\pi = 3.14159265358979323846264 33 \dots .$$

- π as 6-digit floating point decimal machine number, is written as

$$fl(\pi) = 0.314159 \times 10^1$$

1.3 Round-off Errors and Computer Arithmetic

Floating-point form

Any positive real number y can be normalized to the form

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

The floating-point form of y, $fl(y)$ is obtained using two types of floating-point approximations:

- k-**digit Chopping:** all digits $d_{k+1}, d_{k+2}, \ldots$ are chopped off

$$fl(y) = 0.d_1 d_2 \ldots d_{k-1} d_k \times 10^n$$

- k-**digit Rounding:** keep digits up to d_k if $d_{k+1} < 5$ and add one to d_k, if $d_{k+1} \geq 5$:

$$fl(y) = \begin{cases} 0.d_1 d_2 \ldots d_{k-1} d_k \times 10^n & \text{if } 0 \leq d_{k+1} < 5, \quad \text{(down)} \\ 0.d_1 d_2 \ldots d_{k-1}((d_k)+1) \times 10^n & \text{if } 5 \leq d_{k+1} \leq 9, \quad \text{(up)} \end{cases}$$

1.3 Round-off Errors and Computer Arithmetic

Arithmetic using k-digit rounding and chopping

Let x, y be real numbers, let $\circledcirc$ be any of the arithmetic operations

$$+ -, \div, \times$$

The output $x \circledcirc y$ has k-digit floating point form, given as

$$fl(x \circledcirc y) = fl(fl(x) \circledcirc fl(y))$$

- Remember! chop or round at each step of a computation. Rounding (chopping) is not only done on the final output.
- Roundoff errors can propagate through the sequence of computations.

1.3 Round-off Errors and Computer Arithmetic

Example: Arithmetic using k-digit rounding or chopping

Let $x = 0.01234431$, $y = 98.76621$, compute $x + y$ using 4-digit rounding arithmetic.

- Compute $\tilde{x} = fl(x)$ and $\tilde{y} = fl(y)$ to get

$$\tilde{x} = 0.1234 \times 10^{-1}, \quad \tilde{y} = 0.9877 \times 10^{2}$$

- Perform addition in exact arithmetic

$$0.01234 + 98.77 = 98.78234$$

- Take the 4-digit rounding equivalent of the output

$$fl(98.78234) = 0.9878 \times 10^{-2}$$

1.3 Round-off Errors and Computer Arithmetic

Floating-point form (general definition for any base)

Let $\beta \in \mathbb{N}$ and $\beta \geq 2$, a k-digit floating-point form in any base β is of the form

$$\pm(0.d_1 d_2 \ldots d_k)_\beta \times \beta^e$$

where

$$(0.d_1 d_2 \ldots d_k)_\beta = \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_k}{\beta^k}$$

where $d_i \in \{0, 1, 2, \ldots, \beta - 1\}$, $d_1 \neq 0$ and an exponent e

- $\beta = 2$ gives the k-digit binary floating-point form.
- $\beta = 8$ gives the k-digit octal floating-point form.
- $\beta = 10$ gives the k-digit decimal floating-point form.

1.3 Round-off Errors and Computer Arithmetic

Computer Arithmetics (Overflow and Underflow)

k-digit floating-point form in a base β :

$$\pm(0.d_1 d_2 \ldots d_k)_\beta \times \beta^e, \quad m \leq e \leq M$$

For every computing hardware, the bounds m and M on the exponent e represent the range of memory capacity for storing reals: During calculations, if a computed number has an exponent e

- $e > M$ then the memory **overflow** occurs and the device displays `Inf` (∞) or `NaN` (Not a Number).
- $e < m$, then the memory **underflow** occurs and the device considers the number as zero

1.3 Round-off Errors and Computer Arithmetic

Types of Errors

Suppose p^* is an approximation to $p \neq 0$. We define

Actual Error: $p - p^*$ represents the signed difference between the accepted exact value and the approximate value, that is how much your approximation is above or below the exact value.

Absolute Error: $|p - p^*|$ represents the amount of physical error in a measurement, that is, how much different is your approximation to the exact value.

Relative Error: $\dfrac{|p - p^*|}{|p|}$, (provided that $p \neq 0$) represents how good the error in a measurement is, relative to what is being measured, that is, what is the percent difference between your approximation and the exact value.

1.3 Round-off Errors and Computer Arithmetic

Errors as measures of accuracy

Example: Suppose p^* is an approximation to $p \neq 0$.

p	p^*	Absolute error	Relative error
$p = 0.3000 \times 10^1 = 3$	$p^* = 0.3100 \times 10^1 = 3.1$	0.1	$0.333\bar{3} \times 10^{-1}$
$p = 0.3000 \times 10^3 = 300$	$p^* = 0.3100 \times 10^3 = 310$	$0.1 \times 10^2 = 10$	$0.333\bar{3} \times 10^{-1}$
$p = 0.3000 \times 10^4 = 3000$	$p^* = 0.3100 \times 10^4 = 3100$	$0.1 \times 10^3 = 100$	$0.333\bar{3} \times 10^{-1}$

The same relative error, $0.333\bar{3} \times 10^{-1} = 3.3\%$ occurs for widely varying absolute errors.

As a measure of accuracy, the absolute error can be misleading and the relative error is more meaningful since it takes into consideration the size of the value.

1.3 Round-off Errors and Computer Arithmetic

Round-off Errors

Round-off error is produced when approximating a real y by its floating point form $fl(y)$.

Round-off error is computed using absolute error or relative error:

$$|y - fl(y)| \quad \text{or} \quad \frac{|y - fl(y)|}{|y|} \quad \text{when} \quad y \neq 0$$

- Abs. error grows with the size of y, which is one flaw of the floating-point system.
- Relative error produces a smaller error when approximating a real with a float:

1.3 Round-off Errors and Computer Arithmetic

Upper bounds on Round-off Errors

The following error bounds are valid if the base β is even. (e.g $\beta = 10$):

Round-off error bound when using k-digit chopping arithmetic:

$$\frac{|fl_{chopp}(y) - y|}{|y|} \leq 10^{1-k}$$

Round-off error bound when using k-digit rounding arithmetic:

$$\frac{|fl_{round}(y) - y|}{|y|} \leq \frac{1}{2}\, 10^{1-k}$$

Rounding produces half the error when chopping.

Thank You

"Algorithms and Roots of Non-linear Equations"

References:

- Richard L. Burden, J. Douglas Faires, Annette Burden, Numerical Analysis, Brooks/Cole CENGAGE Learning, 9th Edition. ISBN: 978-0-538-73564-3.