# SECTION 1: ROUNDOFF AND TRUNCATION ERRORS

ESC 440 – Numerical Methods for Engineers

# <sup>2</sup> Introduction and Course Overview

#### Mathematical Models

- As engineers, we are interested in
  - Designing and analyzing physical systems
  - Analyzing data
- We can represent these systems or data with mathematical models
  - An equation or system of equations that describe the system or behavior
    - Relates *inputs* to *outputs*
- Mathematical models used for *analysis* and *simulation* 
  - May be done *analytically* by hand
  - More often performed *numerically* on a computer

## **Numerical Analysis**

- In practice, most engineering problems are solved, and analyses are performed *numerically*
  - Using computers
- □ Often, we use simulators
  - Electronic circuits
  - Electromagnetic fields
  - Thermal/fluid systems
  - Structural analysis
- Sometimes, we use existing packages, libraries, toolboxes, functions, etc. E.g.,
  - NumPy, SciPy in Python
  - MATLAB
- Other times, we must write our own code

## **Engineering Analyses**

- Many of the types of analyses we perform as engineers are those you have learned about in previous classes
  - Solution of systems of equations
  - Integration/differentiation
  - Solution of differential equations
  - Optimization
  - Curve fitting, etc.
- You have learned to perform these operations analytically
  - By hand
  - Solutions are *exact*
- In this course, you will learn to solve the same types of problems numerically
  - Using a computer
  - Solutions are *estimates*

### **Numerical Analysis**

- Numerical analysis yields an approximate solution
  - An estimate of the actual solution
- The solution is not exact
  - It includes error
- In this first section of the course, we will learn about *numerical error*
  - Where does it come from?
  - What causes it to increase? Decrease?
  - How do we approximate it?

#### **Course Overview**

- 1. Roundoff & Truncation Error
- 2. Root Finding & Optimization
- 3. Systems of Equations
- 4. Curve Fitting
- 5. Integration
- 6. Ordinary Differential Equations
- 7. Fourier Analysis

# 8 Definitions of Error

#### True Error

Absolute error – the difference between an approximation and the true value

 $E_t = (approx.value) - (true value) = \hat{x} - x$ 

Relative error – the true error as a percentage of the true value

$$\varepsilon_t = \frac{\hat{x} - x}{x} \cdot 100\%$$

Both definitions *require knowledge of the true value!* 

#### Approximating the Error

- 10
- Since we don't know the true value, we can only approximate the error
- Often, approximations are made *iteratively* 
  - Approximate the error as the change in the approximate value from one iteration to the next

$$\widehat{E} = \widehat{x}_{i+1} - \widehat{x}_i$$

#### **Relative Approximate Error**

- 11
- We don't know the true value, so we can't calculate the true error – approximate the error
- Relative approximate error an approximation of the error relative to the approximation itself

$$\varepsilon_a = \frac{approx.error}{approximation} \cdot 100\% = \frac{\hat{E}}{\hat{x}} \cdot 100\%$$

#### **Stopping Criterion**

For iterative approximations, continue to iterate until the relative approximate error magnitude is less than a specified *stopping criterion* 

$$|\varepsilon_a| < \varepsilon_s$$

For accuracy to at least n significant figures set the stopping criterion to

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

# 13 Roundoff Error

### Roundoff Errors

- Roundoff errors occur due to the way in which computers represent numerical values
- Computer representation of numerical values is limited in terms of:
  - Magnitude there are upper and lower bounds on the magnitude of numbers that can be represented
  - *Precision* not all numbers can be represented exactly
- Certain types of mathematical manipulations are more susceptible to roundoff error than others

#### Number Systems – Decimal

#### We are accustomed to the *decimal* number system

- A base-10 number system
- **•** Ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each digit represents an integer power of 10



#### **Binary Number System**

- Computers represent numbers in *binary* format
  - A base-2 number system
  - Two digits: 0, 1
    - Easy to store binary values in computer hardware an on or off switch a high or low voltage
  - One digit is a *bit* eight bits is a *byte*
  - Each bit represents an integer power of 2



#### **IEEE Double-Precision Format**

- 17
- Floating point numbers in Python are represented as 64-bit double-precision floating point values – float64

64-bit binary word



$$\pm \left(1 + \sum_{i=1}^{52} f_i \cdot 2^{-i}\right) \times 2^e$$

### IEEE Double-Precision Format

#### Mantissa

- Only the *fractional portion* of the mantissa stored
- $\blacksquare$  Bit to the left of the binary point assumed to be 1
  - Normalized numbers
  - Really a 53-bit value

#### **Exponent**

- 11-bit signed integer value: -1022 ... 1023
- Two special cases:
  - *e* = 0x000 (i.e. *all zeros*): zero if *f* = 0, *subnormal #'s* if f≠0
  - e = 0x7FF (i.e. **all ones**):  $\infty$  if f = 0, NaN if  $f \neq 0$

#### Normalized numbers

#### Leading zeros are removed

- Most significant digit (must be a 1 in binary) moved to the left of the binary point
- **5**3<sup>rd</sup> bit of the mantissa (always 1) needn't be stored
- Maximum mantissa



### Subnormal Numbers

- 20
- □ If  $f_0 = 1$ , always, then the smallest number that could be represented is:  $2^{-1022} \approx 2.225 \times 10^{-308}$
- □ If we allow for  $f_0 = 0$ , then the most significant bit is somewhere to the right of the binary point
  - Leading zeros not normalized ... subnormal
  - Allows for smaller numbers, filling in the hole around zero
- Subnormal numbers represented by setting the exponent to zero
- Smallest subnormal number:

$$2^{-1022-52} = 2^{-1074} \approx 5 \times 10^{-324}$$

#### Doubles – Range

#### Maximum value:

$$max = \left(1 + \sum_{i=1}^{52} 2^{-i}\right) \times 2^{1023} \approx 1.798 \times 10^{308}$$

#### Minimum normal value:

$$min_{norm} = 2^{-1022} \approx 2.225 \times 10^{-308}$$

#### Minimum subnormal value:

$$min_{sub} = 2^{-1022-52} = 2^{-1074} \approx 5 \times 10^{-324}$$

#### □ Precision – *machine epsilon*

$$\varepsilon = 2^{-52} \approx 2.22 \times 10^{-16}$$

#### Roundoff Error – Mathematical Operations

22

Certain types of mathematical operations are more susceptible to roundoff errors:

- Subtractive cancellation subtracting of two nearlyequal numbers results in a loss of significant digits
- Large computations even if the roundoff error from a single operation is small, the cumulative error from many operations may be significant
- Adding large and small numbers as in an infinite series
- Inner products (i.e., dot product) very common operation solution of linear systems of equations



#### **Truncation Errors**

- Errors that result from *the use of an approximation* in place of an exact mathematical procedure
  - E.g., numerical integration, or the approximation derivatives with finite-difference approximations
- To understand how truncation errors arise, and to gain an understanding of their magnitudes, we'll make use of the *Taylor Series*

#### **Taylor Series**

- Taylor's Theorem any smooth (i.e., continuously differentiable) function can be approximated as a polynomial
- Taylor Series

$$f(x_{i+1}) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_i)}{n!} (x_{i+1} - x_i)^n$$

- This infinite series is an *equality* 
  - An *exact* representation of any smooth function as a polynomial
  - An *infinite-order polynomial* impractical

#### **Taylor Series Approximation**

- 26
- Can approximate a function as a polynomial by *truncating* the Taylor series after a finite number of terms

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

where  $h = x_{i+1} - x_i$  is the **step size** 



#### **Taylor Series Truncation Error**

□ Can account for error by lumping the n + 1 and higher-order terms into a single term,  $R_n$ 

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \dots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

 $\square$   $R_n$  is the error associated with truncating after n terms

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1}$$

 $\Box$   $\xi$  is some (*unknown*) value of x between  $x_i$  and  $x_{i+1}$ 

#### **Derivative Mean-Value Theorem**

□ If f(x) and f'(x) are continuous on  $[x_i, x_{i+1}]$ , then there is a point on this interval,  $\xi$ , where  $f'(\xi)$  is the slope of the line joining  $f(x_i)$  and  $f(x_{i+1})$ 



K. Webb

28

#### Truncation Error – Dependence on Step Size

29

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1}$$

 $\Box$  We don't know  $\xi$ , so we don't know  $R_n$ 

- We do know it's proportional to h<sup>n+1</sup>, where h is the step size
- **\square** Error is **on the order of**  $h^{n+1}$

$$R_n = O(h^{n+1})$$

□ If n = 1 (first-order approx.), *halving* the step size will *quarter* the error

## <sup>30</sup> Truncation Errors in Practice

- Discretizing equations
- Finite-difference approximations

## **Discretization of Equations**

- 31
- As engineers, many of the mathematical expressions we are interested in are differential equations
  - We know how to evaluate derivatives analytically
  - Need an approximation for the derivative operation in order to solve numerically
- Discretization conversion of a continuous function, e.g., differentiation, to a discrete approximation for numerical evaluation

## Finite Difference Approximations

32

Recall the definition of a derivative

$$f'(x_i) = \lim_{h \to 0} \frac{f(x_{i+1}) - f(x_i)}{h}$$

Remove the limit to approximate this numerically

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

This is the *forward difference approximation* 

Uses value at x<sub>i</sub> and forward one step at x<sub>i+1</sub> to approximate the derivative at x<sub>i</sub>

#### Discretizing Equations – Example

A free-falling object can be modeled as

$$\frac{dv}{dt} = g - \frac{c_d}{m}v^2$$

where v is velocity, m is mass, g is gravitational acceleration, and  $c_d$  is a lumped drag coefficient

 This is a non-linear ordinary differential equation (ODE), which can be solved analytically to yield

$$v(t) = \sqrt{\frac{mg}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} \cdot t\right)$$

#### **Discretizing Equations – Example**

To solve numerically instead, approximate the derivative operation with a finite difference

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \cong g - \frac{c_d}{m} v(t_i)^2$$

Solving for v(t<sub>i+1</sub>) and using h to denote the time step yields

$$v(t_{i+1}) \cong v(t_i) + \left[g - \frac{c_d}{m}v(t_i)^2\right]h$$

- We've transformed the *differential equation* to a *difference equation* 
  - An *algebraic* equation
  - Can be solved iteratively using a loop

34

#### **Discretizing Equations – Example**

$$v(t_{i+1}) \cong v(t_i) + \left[g - \frac{c_d}{m}v(t_i)^2\right]h$$

- The term in the square brackets is the original diff. eq., i.e., it is v'(t)
- The difference equation is a *first-order Taylor series* approximation

$$v(t_{i+1}) = v(t_i) + v'(t_i) h + R_1$$

 Where we know that the error is on the order of the step size squared

$$R_1 = O(h^2)$$

 Taylor series provides a relation between the step size and the accuracy of the numerical solution to the diff. eqn.

35

## Finite Difference Methods

- The preceding example showed
  - One method forward difference for numerically approximating a derivative
  - Transformation of a differential equation to a difference equation
  - How Taylor series can provide an understanding of the error associated with an approximation
- Now we'll take a closer look at three *finite difference methods* and how Taylor series can help us understand the error associated with each

#### **Forward Difference**

 Can also derive the forward difference approximation from the Taylor Series

 $f(x_{i+1}) = f(x_i) + f'(x_i)h + R_1$ 

□ Solving for  $f'(x_i)$ 

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{R_1}{h}$$

We've already seen that

 $R_1 = O(h^2)$ 

□ So, the error term is

$$\frac{R_1}{h} = O(h)$$

 The forward difference, including error, is

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

 Error of the forward difference approximation is on the order of the step size

#### **Forward Difference**



Value of the function, f(x), at  $x_i$  and forward one step at  $x_{i+1}$ used to approximate the derivative at  $x_i$ 

#### **Backward Difference**

 Backward difference uses value of f(x) at x<sub>i</sub> and one step backward at x<sub>i-1</sub> to approximate the derivative at x<sub>i</sub>

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

 This can also be developed by expanding the Taylor series backward

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + R_1$$

• Then solving for  $f'(x_i)$ 

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \frac{R_1}{h}$$

 Again the error is on the order of the step size

$$\frac{R_1}{h} = O(h)$$

 The backward difference expression, including error, becomes

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

Error of the backward
difference approximation is
on the order of the step size

#### **Backward Difference**

40



Now use the value of f(x)at  $x_i$  and backward one step at  $x_{i-1}$  to approximate the derivative at  $x_i$ Again, error is

R = O(h)

### **Central Difference**

Central difference uses value of f(x)
one step backward at x<sub>i-1</sub> and ones
step ahead at x<sub>i+1</sub> to approximate
the derivative at x<sub>i</sub>

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

- This can also be developed by subtracting the backward Taylor series from the forward series
- Second-order derivative terms cancel, leaving

$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + R_2$$

Now, the remainder term is

$$R_2 = \mathcal{O}(h^3)$$

 The central difference expression, including error, becomes

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

- Error of the central difference approximation is on the order of the step size <u>squared</u>
- Central difference method is more accurate than forward or backward
  - Uses more information

#### **Central Difference**

42



Now use the value of f(x)backward one step at  $x_{i-1}$  and forward one step at  $x_{i+1}$  to approximate the derivative at  $x_i$  Reduced error:  $R = O(h^2)$ 

# 43 Total Numerical Error

## **Total Numerical Error**

#### Total numerical error is the sum of roundoff and truncation error

- Roundoff error is largely out of your control, and, with double precision arithmetic, it is not typically an issue
- Truncation error can be a significant problem, but can be reduced by decreasing step size
- Reducing step size reduces truncation error, but may also result in subtractive cancellation, thereby increasing roundoff error
- Choose step size to minimize total error
  - Or, more typically, to reduce truncation error to an acceptable level

## **Total Numerical Error**

- 45
- Reducing step size reduces truncation error, but may also result in subtractive cancellation, thereby increasing roundoff error
- Could choose step size to minimize total error
- But, more typically, reduce step size just enough to reduce truncation error to an acceptable level



## Central Difference Error Analysis

- 46
- First derivative of a function in terms of the central difference approximation is

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(\xi)}{6}h^2$$

The last term on the right is the truncation error
There is also roundoff error associated with each value

$$f(x_{i-1}) = \tilde{f}(x_{i-1}) - e_{i-1}$$
$$f(x_{i+1}) = \tilde{f}(x_{i+1}) - e_{i+1}$$

were  $\tilde{f}(x_i)$  represents a rounded value, and  $e_i$  is the corresponding roundoff error

## Central Difference Error Analysis

- 47
- Substituting the expressions for the rounded values into the expression for the true derivative yields

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(\xi)}{6}h^2 + \frac{e_{i+1} - e_{i-1}}{2h}$$

Giving a total error of



- Truncation error increases with step size
- Roundoff error decreases with step size