Why Study Numerical Methods

- Numerical methods are extremely powerful problem-solving tools. They are capable of handling large system of equations, non-linearities, and complicated geometries that are common in engineering practice, and that are often impossible to solve analytically. As such, they greatly enhance our problem-solving skills.

- During your engineering careers, you may often have occasions to use commercially available “pre-packaged”, or “canned” computer programs that involve numerical methods. The intelligent use of these programs often is predicated on knowledge of the basic theory underlying the methods.

- Many problems cannot be approached using canned programs. If you are conversant with numerical methods and are adept at computer programming you will have the capability of designing your own programs to solve problems without having to buy or commission expensive software.

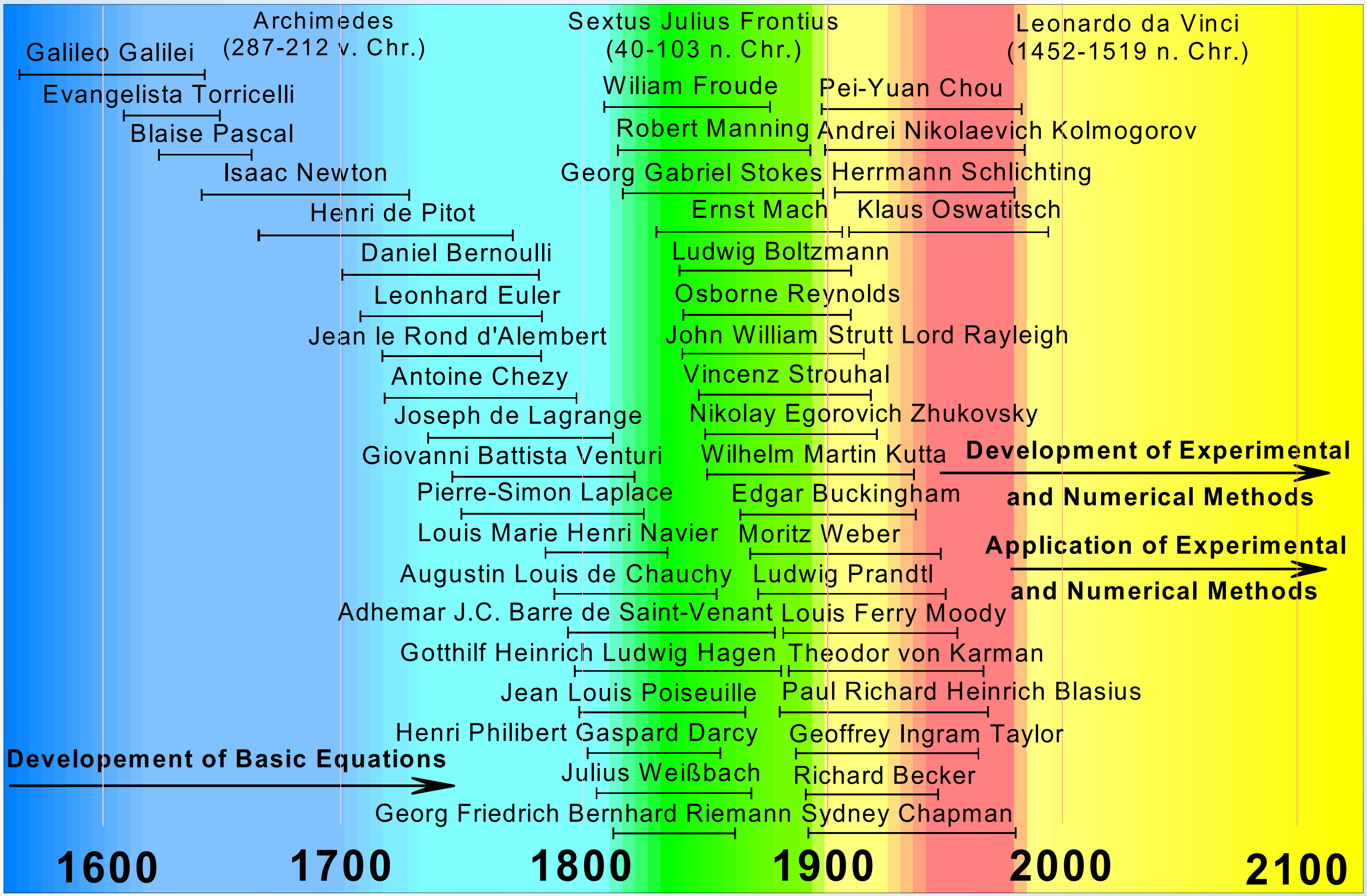
- Numerical methods are an efficient vehicle for learning to use computers. It is well known that an effective way to learn programming is to actually write computer programs. Because numerical methods are for the most part designed for implementation on computers, they are ideal for this purpose. Further, they are especially well suited to illustrate power and the limitations of computers. When you successfully implement numerical methods on a computer and then apply them to solve otherwise intractable problems, you will be provided with a dramatic demonstration of how computers can serve your professional development. At the same time, you will also learn to acknowledge and control the errors of approximation that are part and parcel of large-scale numerical calculations.

- Numerical methods provide a vehicle for you to reinforce your understanding of mathematics. Because one function of numerical methods is to reduce higher mathematics to basic arithmetic operations, they get at the 'nuts and bolts' of some otherwise obscure topics. Enhanced understanding and insight can result from this alternative perspective.

See the pdf file «Engineering Mathematicians» on «OdtuClass».

See the pdf file «A Brief History of Computing» on «OdtuClass».

ME – 510 NUMERICAL METHODS FOR ME II



REVIEW OF ME-310

Given	Complicated f(x) or Tabulated data with no statistical error
Problem	Represent f(x) or the table with a *simple* function, g(x)
Objective	Operate on g(x) instead of f(x) or the tabulated values
Operations	At a given x or several x's, find • functional values (interpolation) • derivatives • integrals • estimated errors • etc
Choice of g(x)	• Polynomials • Fourier series (Sines and/or Cosines) • Any other *suitable*, *simple* function

Jean Baptiste Joseph Fourier

French Mathematician

1768 – 1830

REVIEW OF ME-310

Basic Theorem: **Taylor Series** (expansion)

$$f(x) = \underbrace{f(x_0) + (x - x_0)\, f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \ldots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0)}_{g(x)} + \underset{\text{Remainder}}{R}$$

$$f(x) = g(x) + R$$

x_0 is the point of expansion

R is the error of truncation (discretization)

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

$$\xi = ?$$

Brook Taylor, FRS

English Mathematician

1685 - 1731

FRS : Fellow of the Royal Society

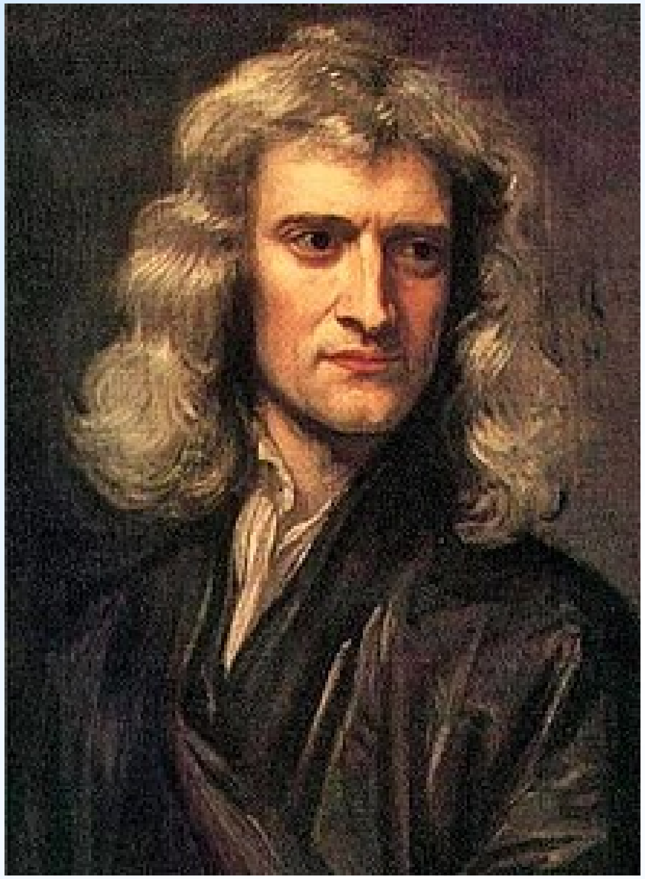
REVIEW OF ME-310

Example: Polynomial fitting

to a set of data points using Newton-Gregory forward polynomials

- Find $P_n(x)$, n-degree polynomial, exactly passing through a given set of equally-spaced data points,
- $x_0, x_1, \ldots, x_n$, $h = \Delta x =$ constant, and $s = \dfrac{x - x_0}{h}$
- Remember definition of forward differences, Δf , $\Delta^2 f$, etc.

Distiguish between «order» and «degree».



Isaac
Newton
1642 - 1726

James Gregory
1638 - 1675

REVIEW OF ME-310

$$g(x) = P_n(x) = f_0 + s\,\Delta f_0 + \frac{s\,(s-1)}{2!}\,\Delta^2 f_0 + \ldots + \frac{s\,(s-1)\,(s-2)\,\ldots\,(s-n+1)}{n!}\,\Delta^n f_0$$

Error:

$$g(x) = E(x) = f(x) - P_n(x)$$

$$= \frac{s\,(s-1)\,(s-2)\,\ldots\,(s-n+1)}{n!}\,\Delta^n f_0$$

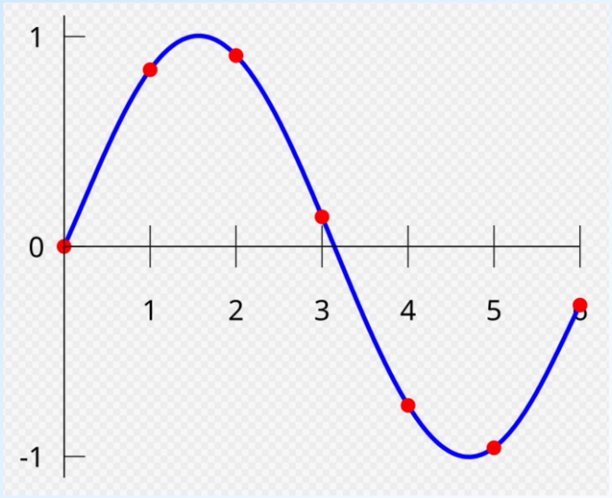
REVIEW OF ME-310

Example: Given a complicated f(x) in [a , b] what is the *best-fitting*, single polynomial of small degree, n? To do what? Error?

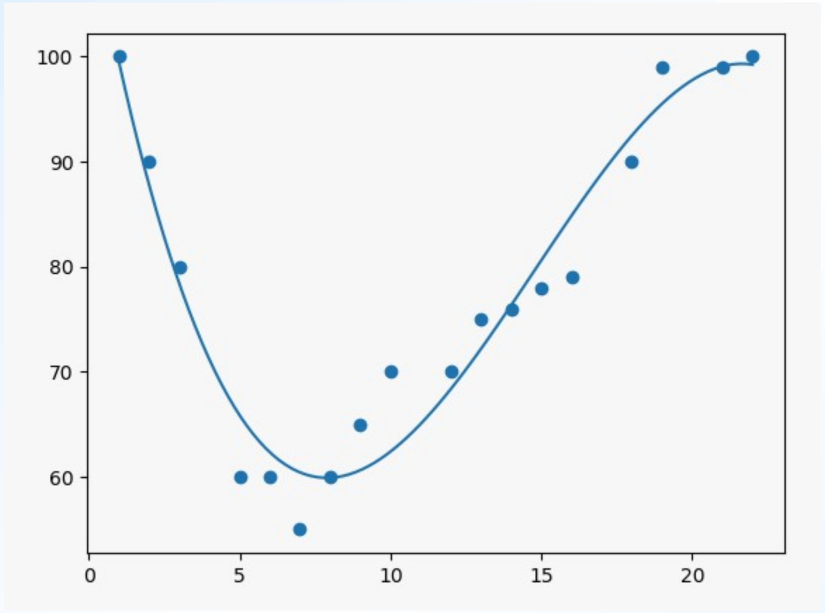
Example: Given a set of data points (not a function), how do you find the representing polynomial of degree n, $P_n(x)$? To do what? Error?

Example: Given a set of data points (not a function), how do you find the polynomial of degree n, $P_n(x)$, that passes through the given number of data points, exactly? To do what? Error?

Example: For a large number of points, remember *cubic splines*. To do what? Error?



Polynomial (or something else) that passes through a given number of data points (for interpolation?)



Polynomial (or something else) that represents the given set of data points (for what?)

COMPUTATIONAL PROCEDURE

Frequently, several methods are available for the numerical solution of a given mathematical problem. Few relevant criteria for selection of a method:

- Accuracy
- Efficiency
- Numerical stability
- Programming simplicity
- Versatility
- Computer storage requirements
- Interfacing with available software
- Previous experience with a given method

SOME PITFALLS IN NUMERICAL COMPUTATION

* Errors in the input data
  - Systematic (deterministic) errors or temporary disturbances during measurements
  - Round-off errors (√2 or π are shortened)
* Conversion errors (0.1 requires infinite number of bits to be exactly represented in base-2)
* Round-off errors during computation
* Truncation (discretization) errors
* Simplifications in the physical mathematical models
* Errors due to the chosen numerical method
* Errors due to the algorithm used (such as catastrophic cancellation)

APPROXIMATIONS

Computing Surface area of the Earth using the formula

$$A = 4 \pi r^2$$

involves several approximations:

* Earth is modeled as a sphere, an idealization of its true shape;

* Value for radius r is based on empirical measurements and previous computations;

* Value for π requires truncating an infinite process;

* Values for input data and results of arithmetic operations are rounded in computer.

ROUND-OFF ERROR DURING COMPUTATIONS

Find E using a computter

$$E = 10^6 \left[\left(\frac{1}{27} \right)^3 - \left(\frac{25}{27} - \frac{8}{9} \right)^3 \right]^{1/3}$$

Note that the exact answer is zero. Follow this procedure:

- Store $A = \left(\frac{1}{27} \right)^3$ $B = \left(\frac{25}{27} \right)$ $C = \frac{8}{9}$
- Calculate and store $D = (B - C)^3$
- Calculate, store and print $E = 10^6 \, (A - D)^{1/3}$

ROUND-OFF ERROR DURING COMPUTATIONS

The answer, with a PC and Fortran compiler, is:

Single precision:	E = 443.632800
Double precision:	E = 0.00000000

CATASTROPHIC CANCELLATION

Problem: Calculate e^x using n number of terms in the MacLaurin Series

$$S_n = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^{n-1}}{(n-1)!}$$

and find what n should be if S_n is to be correct to 14 digits after the decimal point, i.e., the truncation error is less than $0.5\ 10^{-14}$.

Answer: n = 22 when x = 2.
n = 77 when x = 25.
n = ?? when x = - 25.

The computer program gives an odd and erronous number when the terms in the series are added in the above order without paying attention that the sign of successive terms are alternating which causes catastrophic cancellation when x is large enough.

ABSOLUTE ERROR AND RELATIVE ERROR

Absolute Error = | True Value - Approximate Value |

Relative (percent) Error = | Absolute error / True Value | * 100

Equivalently, Approximate Value = (True Value) (1 + Relative error)

True Value is usually unknown. So, we estimate or bound error rather than compute it exactly.

For the same reason, relative error is often taken to be relative to approximate value, rather than true value.

ERRORS

True Error => usually not known $E_{true}(x) = f(x) - P_n(x)$

Estimated Error => approximated

Approximate Error => estimated

$$\left.\begin{array}{l}\text{Estimated Error}\\ \text{Approximate Error}\end{array}\right\} \quad E_{est}(x) \cong R$$

True Relative Error => relative to the true value $$E_{true}(x) = \frac{f(x) - P_n(x)}{f(x)}$$

Estimated Relative Error => relative to the approximate value $$E_{est}(x) \cong \frac{R}{P_n(x)}$$

ERRORS

Truncation Error => series representation

Round-off Error => lost precision

Deterministic Error => due the measuring instrument

Statistical (stocastic) Error => result of experiments

ERRORS

Programming Errors => **Syntax error (no compilation)**

=> **Run-time error (compiles, but stops running)**

=> **Logical error (compiles and runs, but the output is wrong)**

Read the article «Errors and Error Estimation» on «OdtuClass».

SENSITIVITY and CONDITIONING

Problem is *insensitive*, or *well-conditioned*, if given relative change in input causes commensurate relative change in solution.

Problem is *sensitive*, or *ill-conditioned*, if relative change in solution can be much larger than that in input data.

$$\text{Condition Number} = \frac{|\text{Relative Change in solution}|}{|\text{Relative change in input data}|} = \frac{\left[f(\bar{x}) - f(x) \right] / f(x)}{(\bar{x} - x) / x}$$

Problem is *sensitive* or *ill-conditioned* if Condition Number >> 1

EXAMPLE: SENSITIVITY

Consider the problem of computing cosine function for arguments near $\pi/2$.

Let $x \approx \pi/2$ and let h be a small perturbation to x.

Then, the error in $\cos(x + h)$ is given by

$$\text{Absolute Error} = \cos(x + h) - \cos(x) \approx -h\sin(x) \approx -h$$

$$\text{Relative Error} \approx -h\tan(x) \approx \infty$$

So, small changes in x near $\pi/2$ cause large relative changes in $\cos(x)$ regardless of the method used for computing it.

Example: $\cos(1.57079) = 0.63267949\ 10^{-5}$

$\cos(1.57078) = 1.63267949\ 10^{-5}$

Relative change in output is about a quarter million times greater than relative change in input.

EXAMPLE: EVALUATING FUNCTION

When function f is evaluated for approximate input argument $x + h$ instead of true input value x

$$\text{Absolute Error} = f(x+h) - f(x) \approx h\, f'(x)$$

$$\text{Relative Error} = \frac{f(x+h) - f(x)}{f(x)} \approx h\, \frac{f'(x)}{f(x)}$$

$$\text{Condition Number} = \frac{h\, f'(x)\,/\,f(x)}{h\,/\,x} \approx x\, \frac{f'(x)}{f(x)}$$

Relative error in function value can be much larger or smaller than that in input.

STABILITY AND ACCURACY

Stability of an algorithm is analogous to conditioning of a problem.

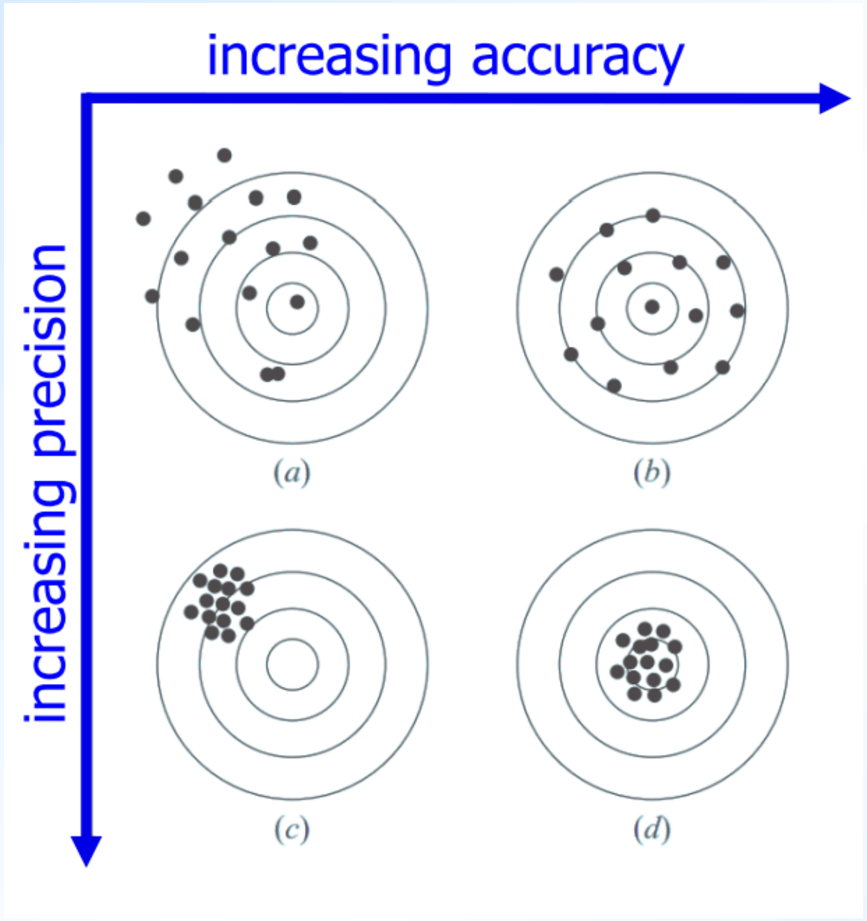
Algorithm is *stable* if result is relatively insensitive to perturbations due to approximations made during computation.

Accuracy refers to closeness of computed solution to true solution of problem.

Accuracy depends on conditioning of problem as well as stability of algorithm.

Inaccuracy can result from applying stable algorithm to ill-conditioned problem, or applying unstable algorithm to well-conditioned problem.

ACCURACY and PRECISION



ACCURACY

How closely computed or measured values agree with the true value

PRECISION

How closely computed or measured values agree with each other