MA2501: Numerical Methods

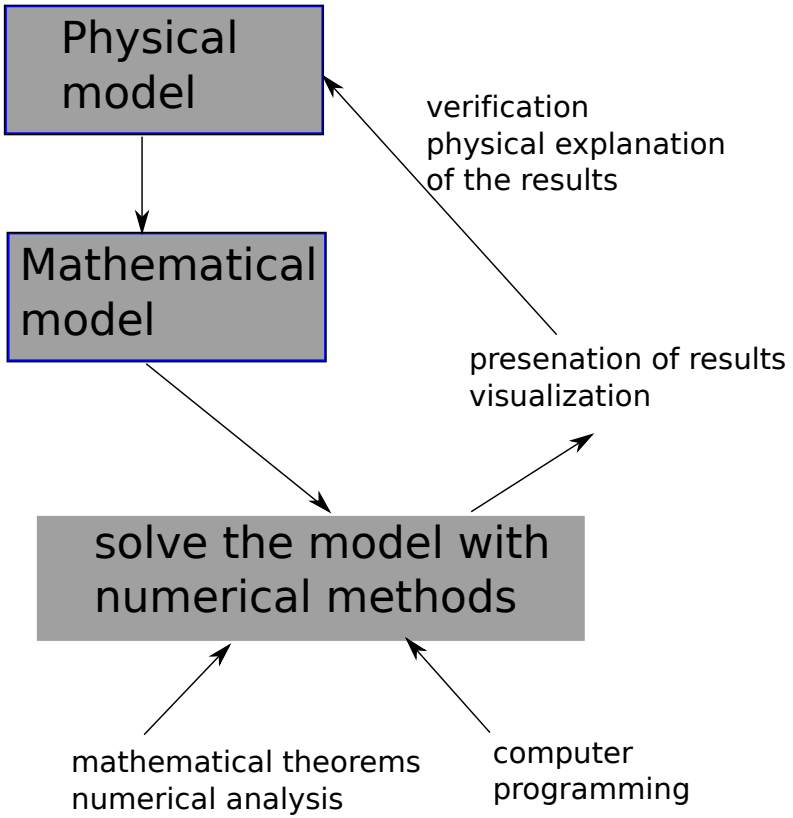
Asif Mushtaq

Department of Mathematical Sciences NTNU, Trondheim
Email: asifm@math.ntnu.no
Office: 1301

January 11, 2016

What are numerical methods?

- Algorithms that compute approximations to the solutions of the equations or similar things
- Bigger picture:



Mathematical Preliminaries!

Taylor Series of function about a point

The formal Taylor series of f about c is

$$f(x) = f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \frac{f'''(c)}{3!}(x-c)^3 + \dots$$

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k \tag{1}$$

The special case when $c = 0$, known as *Maclaurin series*

$$f(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!}(x)^2 + \frac{f'''(0)}{3!}(x)^3 + \dots$$

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}(x)^k \tag{2}$$

Some Familier Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sum_{k=0}^{\infty} (-1)^k \frac{(x)^{2k+1}}{(2k+1)!}$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{2k!}$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots = \sum_{k=0}^{\infty} x^k \quad (x \neq 1)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots = \sum_{k=1}^{\infty} (-1)^{(k-1)} \frac{x^k}{k}$$

(3)

Examples

Use Taylor series to find ln2. Since

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, \tag{4}$$

take $x = 1$, we have

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} \ldots \tag{5}$$

* Add the first eight terms, we have

$$\ln 2 \approx 0.63352 \qquad \text{(poor approximation)}$$
$$\ln 2 \approx 0.69315\ldots \qquad \text{(exact value)}$$

Continue...

Use Taylor a different series. Since

$$\ln\left(\frac{1+x}{1-x}\right) = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots\right). \tag{6}$$

* Take $x = \frac{1}{3}$, add the first four terms and multiply by 2, we have

$$\ln 2 \approx 0.69313 \qquad \text{(good approximation)}$$
$$\ln 2 \approx 0.69315\dots \qquad \text{(exact value)}$$

$\ln\left(\frac{1+x}{1-x}\right)$ at $\frac{1}{3}$ is near point of expansion.
We are exploiting a more rapidly convergent series!

Continue...

Rule

Rapid convergence of a Taylor series can be expected near the point of expansion, but not at remote points!

Remark: Significant digits are digits beginning with the leftmost nonzero digit and ending with right most correct digit, including final zeros !

Significant Digits of Precision

Explain with example

Consider following system of linear equation (Detail discussion in CH–2). Solve the system for variable y.

$$0.1036x + 0.2122y = 0.7381$$
$$0.2081x + 0.4247y = 0.9327$$

- First, carry only three significant digits of precision.
- Second, repeat with four significant digits of precision.

With three significant digits :

- Round all the numbers in the original problem to three digits.

$$0.104x + 0.212y = 0.738$$
$$0.208x + 0.425y = 0.933$$

- Solve system by choosing $\alpha = \frac{0.208}{0.104} \approx 2.000$.
- $\implies y = -543.000$.

With four significant digits :

- Take the original problem without rounding the number
- Solve system by choosing $\alpha = \frac{0.2081}{0.1036} \approx 2.0087$.
- $\implies y = 355.66$.

Alert!!!, Very huge difference in answer with slight change in significant of digits

- **In summary,**

$$\begin{aligned}
&\Longrightarrow \quad y = -543.000 \qquad \text{(3 significant of digits)} \\
&\Longrightarrow \quad y = 355.66 \qquad \text{(4 significant of digits)} \\
&\Longrightarrow \quad y = 356.2907199 \qquad \text{(10 significant of digits)} \\
&\Longrightarrow \quad y = 356.2907442151 \qquad \text{(MATLAB)}
\end{aligned}$$

- The lesson learned is that data thought to be accurate should be
  - carried with full precision
  - not rounded prior to each calculations

Alert!!! But all calculator and computers have limited precision!

QUESTION:What to do?

Computer Arithmetic

- Computers use **binary arithmetic,** representing each number as a binary number (a finite sum of integer powers of 2)
- Some numbers can be represented exactly, but others, such as $\frac{1}{10}, \frac{1}{100}, \frac{1}{1000}, \ldots$ cannot.
  For example,
  $$2.125 = 2^1 + 2^{-3}$$
  has an exact representation in binary (base 2), but
  $$3.1 \approx 2^1 + 2^0 + 2^{-4} + 2^{-5} + 2^{-8} + \ldots$$
  does not.
  And, of course, there are the transcendental numbers like π that have no finite representation in either decimal or binary number system.

Floating-Point Representation

Computers can store **integers** exactly but not **real numbers** (in general). Suppose $x \neq 0$ is a real number, then normalized floating-point representation:

- **In decimal system**

$$x = \pm 0.d_1 d_2 d_3 \ldots \times 10^n$$

$$x = \pm r \times 10^n, \quad \left(\frac{1}{10} \leq r < 1\right)$$

where, r is **normalized mantissa** and n the **exponent**

- **In binary system**

$$x = \pm q \times 2^m, \quad \left(\frac{1}{2} \leq q < 1\right)$$

where, q **mantissa** would be expressed as a sequence of zeros or ones in the form $q = (0.b_1 b_2 b_3 \ldots)_2$.
Further $b_1 \neq 0 \implies b_1 = 1$

Floating point model

Computers have finite memory, hence not every number can be represented exactly on a computer.
Examples: $\sqrt{2}$, π have a infinite number of digits.

- in a computer, a fixed amount of memory is allocated to represent a number:

$$fl(x) = \pm(0.b_k b_{k-1} \ldots b_{k-r-1}\, \tilde{b}_{k-r}) \cdot 2^m$$

Floating point model

- Single precision IEEE-754[1] (32 bits)

- Double precision IEEE-754 (64 bits)

- Long double precision IEEE-754 (80 bits)

[1] WIKIPEDIA: The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). Many hardware floating point units use the IEEE 754 standard.

Chopping and rounding off

There are two ways to terminate the mantissa of the kth decimal machine number approximating $0.d_1d_2d_3\ldots d_kd_{k+1}d_{k+2}\ldots\times 10^n$, $0\leq d_i\leq 9, d_1\neq 0$

- **Chopping:** Chop off the digits $d_{k+1}d_{k+2},\ldots$ to get $0.d_1d_2d_3\ldots d_k\times 10^n$,.
- **Rounding:** Add $5\times 10^{n-(k+1)}$ and chop off the $k+1,k+2,\ldots$ digits (if $d_{k+1}\geq 5$ we add 1 to d_k before chopping)

Rounding is more accurate than chopping.

Example: $\pi = 3.14159265359\ldots$
Five digits floating point form by using chopping is $\pi = 0.31415\times 10^1$ and with rounding $\pi = 0.31416\times 10^1$

Measure of the error: Absolute and Relative

Suppose β be the approximation to α. We define two measures of the error

- **Absolute error:** $E = |\alpha - \beta|$
- **Relative error:** $E_r = \frac{|\alpha-\beta|}{|\alpha|}$
  Example:
  - If $\alpha = 2$ and $\beta = 2.1$ (absolute error $E = 10^{-1}$)
  - If $\alpha = 2 \times 10^{-3}$ and $\beta = 2.1 \times 10^{-3}$
  (absolute error $E = 10^{-4}$)
  - If $\alpha = 2 \times 10^{3}$ and $\beta = 2.1 \times 10^{3}$ (absolute error $E = 10^{2}$)

But in all three cases the relative error remains same,
$E_r = 5 \times 10^{-2}$.

Horner´s Algorithm

Other names: Nested multiplication, Synthetic Division

- To evaluate the polynomial,

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + + a_{n-1} x^{n-1} + a_n x^n$$

group the terms in a nested multiplication like

$$p(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + x(a_n))))$$

Horner´s algorithm can evaluate a polynomial with only n additions and n multiplications.
This is the minimum number of operations possible.

Example

Other names: Nested multiplication, Synthetic Division

- The Horner form the polynomial,

$$p(x) = 5 + 3x - 7x^2 + 2x^3$$

is,

$$p(x) = 5 + x(3 + x(-7 + x(2)))$$

for a given value of x.

We have avoided all the exponentiation operations by using Horner´s algorithm.

Thank you!