MATH 151 A: Applied Numerical Methods I

Will learn:

1) How to represent numbers and do arithmetic on computer, and track/control their errors (Chpt 1)
2) Algorithms to solve 1D non-linear eqns (Chpt 2) and linear systems (Chpt 6)
3) Numerical Implementation/Approximation:
   - Polynomial Interpolation (Chpt 3)
   - Numerical Differentiation/Integration (Chpt 4)

Will focus on 3 aspects of algorithms:

1) **Recipe**: description of methods
2) **Analysis**: accuracy, stability, and computational complexity
3) Underlying mathematics

Errors in mathematical computation

- finite precision arithmetic used by computers (e.g. round-off errors)
- discretization or truncation errors (e.g. estimating an integral by sum)

- modeling errors (e.g. simplifying assumptions)
- measurement or data collection errors
- electronic errors (e.g. random bit flips)

Scientific Machine Numbers (Normalized)

$$\pm\, \underbrace{0.\, d_1 d_2 \cdots d_n \cdots}_{\text{significand or mantissa}} * \underset{\text{base}}{b}^{\overset{\text{exponent}}{q}}$$

where $d_1 \neq 0$ and $0 \leq d_i < b$

Common Choices for b: $\underset{\text{decimal}}{10}$, $\underset{\text{binary}}{2}$, $\underset{\text{hexadecimal}}{16}$

Note: computer cannot store an infinite number of digits for significand, so numbers can be represented as floating point numbers

Floating Point Numbers

$$\pm\, 0.\, d_1 d_2 \cdots d_n * b^q$$

where $d_1 \neq 0$, $0 \leq d_i < b$, $q_{min} \leq q \leq q_{max}$

Ex 1: Normalized scientific notation for π

$$\pi = 0.3141592\ldots * 10^1$$

Normalized floating point representation with $n=6$, $b=10$

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

Ex 2: Decimal vs. Binary Representation

$$(101.01)_2 = 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2}$$

base

$$= (5.25)_{10}$$

base 10

IEEE Floating Point Standard

Numbers are represented using specific number of bits (0 and 1s)

Single-Precision (base $b=2$, use 32 bits)

±	Exponent	Significand
1 bit	8 bits	23 bits

This can represent numbers $\approx 10^{-38}$ to 10^{38}

Double Precision (base $b=2$, use 64 bits)

±	Exponent	significand
1	11 bits	52 bits

This can represent numbers $\approx 10^{-307}$ to 10^{307}

MATLAB Default

Note: Floating Point Numbers have limitations:

- limitations on significand
- limitations on exponent
  - ↳ overflow ⇒ exponent is too large (positive)
  - underflow ⇒ exponent is too large (negative)

Overflow is set as "Inf"
Underflow is set as zero

Remark: with proper scaling, overflow/underflow can often be avoided

1.2 Errors

Suppose $p^* \in \mathbb{R}$ is an approx. of $p \in \mathbb{R}$

- absolute error: $e_a(p, p^*) = |p - p^*|$
- relative error: $e_r(p, p^*) = \dfrac{|p - p^*|}{|p|}$ for $p \neq 0$

Error Bounds

- Absolute error bound: $e_a(p, p^*) \leq \varepsilon_a(p, p^*)$
- Relative error bound: $e_r(p, p^*) \leq \varepsilon_r(p, p^*)$

Remark: often we can only obtain a bound on error produced by algorithms.

Ways to Reduce Errors in Finite Digit Precision

I) Avoid subtraction of 2 nearly equal numbers

Reason: causes cancellation of significant digits (catastrophic cancellation)

Ex 1: Given 2 numbers, x and y, with $x > y$ and k-digit representation

Then

$$fl(x) = 0.d_1 d_2 \dots d_p \alpha_{p+1} \alpha_{p+2} \dots \alpha_k \times 10^n$$

$$fl(y) = 0.d_1 d_2 \dots d_p \beta_{p+1} \beta_{p+2} \dots \beta_k \times 10^n$$

$$\Rightarrow fl(fl(x) - fl(y)) = 0.\underbrace{00\dots0}_{p \text{ times}} \sigma_{p+1} \sigma_{p+2} \dots \sigma_k \times 10^n$$

$$= 0\,\sigma_{p+1} \sigma_{p+2} \dots \sigma_k \times 10^{n-p}$$

$\Rightarrow fl(fl(x) - fl(y))$ has $k-p$ significant digits (loss of accuracy)

Ex 2: $f(x) = \dfrac{1-\cos(x)}{x^2}$

Fact: $0 < f(x) \leq \frac{1}{2}$, $\lim_{x \to 0} f(x) = \lim_{x \to 0} \dfrac{\sin(x)}{2x}$ L'Hospital's

$$= \lim_{x \to 0} \frac{\cos(x)}{2} = \frac{1}{2}$$

ON MATLAB:

$f(1.2 * 10^{-8}) \approx 0.77098$

$f(1.2 * 10^{-9}) = 0$ NOT $\approx \frac{1}{2}$!

Ex 2: $f(x) = \dfrac{1-\cos(x)}{x^2}$

L'Hospital's

Fact: $0 < |f(x)| \le \frac{1}{2}$, $\lim\limits_{x\to 0} f(x) = \lim\limits_{x\to 0} \dfrac{\sin(x)}{2x}$

ON MATLAB: $= \lim\limits_{x\to 0} \dfrac{\cos(x)}{2} = \frac{1}{2}$

$f(1.2 * 10^{-8}) \approx 0.77098$

$f(1.2 * 10^{-9}) = 0$ NOT $\approx \frac{1}{2}$!

Remedy (For this problem)

Use trig identity $\cos(x) = 1 - 2\sin^2\left(\frac{x}{2}\right)$ to get

$$f(x) = \frac{1-\cos(x)}{x^2} = \frac{2\sin^2\left(\frac{x}{2}\right)}{x^2}$$

II) Avoid Division by Small Number or Multiplying by large numbers Reason: Avoid Overflow

Ex: Consider $c = \sqrt{a^2 + b^2}$

If $a = 10^{170}$, $b = 1$, then correctly rounded sol.: $c = 10^{170}$

However, with double precision arithmetic:

$a^2 = \text{Inf}$, $a^2 + b^2 = \text{Inf} + 1 = \text{Inf}$, $c = \sqrt{\text{Inf}} = \text{Inf}$

Remedy: Scale the data

$$c = s\sqrt{\left(\frac{a}{s}\right)^2 + \left(\frac{b}{s}\right)^2}, \quad \text{where } s = \max\{|a|, |b|\}$$

Here, $s = 10^{170}$, and $c = 10^{170}\sqrt{(1)^2 + \left(\frac{b}{s}\right)^2} = 10^{170}$

$\left(\frac{b}{s}\right)^2 \hookrightarrow$ underflow to 0

III) Reduce the number of arithmetic computations

Reason: more computations $\Rightarrow$ more rounding errors

Ex: Evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ at $x = 4.71$

mult. 2 2 1 = 5

add/sub. 1 1 1 = 3

} 8

Now consider nested formulation:

$$f(x) = (x^3 - 6.1x^2 + 3.2x) + 1.5$$
$$= (x^2 - 6.1x + 3.2)x + 1.5$$
$$= ((x - 6.1)x + 3.2)x + 1.5$$

2 multiplications

3 additions

(will see later in Chpt 2 Horner's Method)

Truncation Error: (truncate infinite sum by finite sum)

Taylor's Thm:

$$f(x) = \underbrace{f(x^*) + f'(x^*)(x - x^*) + \frac{f''(x^*)}{2!}(x - x^*)^2 + \cdots + \frac{f^{(n)}(x^*)}{n!}(x - x^*)^n}_{P_n(x)}$$

$$+ \underbrace{\frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x^*)^{n+1}}_{R_n(x)} \quad \text{where } \xi \text{ between } x \text{ and } x^*$$

Truncation error $R_n(x) = f(x) - P_n(x)$

1.3 Algorithms and Convergence

Algorithm: procedure that unambiguously describes a finite sequence of steps in a specified order
(can typically be written as pseudocode $\Rightarrow$ no specific language required)

→ **stable:** small changes to input $\Rightarrow$ small changes to output

conditionally stable: stable for some input

unstable: not stable for any input

Let $E_0 > 0$ be error at initial step
E_n be error at n^{th} step

Algorithm has

- **linear error growth:** if $E_n \approx C \cdot n \cdot E_0$, $C > 0$ constant
- **exponential error growth:** If $E_n \approx C^n \cdot E_0$, $C > 1$ constant

Remark: linear error growth $\Rightarrow$ stable
exponential error growth $\Rightarrow$ unstable

Convergence Rate of Sequences

Let $\{\alpha_n\}_{n=1}^{\infty}$ be a seq. s.t.

$$\alpha_n \to \alpha \quad \text{as} \quad n \to \infty.$$

Q: How fast is α_n approaching α?

Use a second known seq. $\{\beta_n\}$ to describe convergence behavior of $\{\alpha_n\}$.

Def: Let $\alpha_n \to \alpha$ and $\beta_n \to 0$ as $n \to \infty$.
If there exists $K > 0$ and integer n_0 such that

$$|\alpha_n - \alpha| \le K |\beta_n| \quad \text{for all} \quad n \ge n_0,$$

Then α_n converges to α with rate/order of $O(\beta_n)$, written as $\alpha_n = \alpha + O(\beta_n)$

Remark 1: β_n is usually chosen as n^{-p} $\left(\frac{1}{n^p}\right)$, $p > 0$

Generally interested in largest possible p s.t.

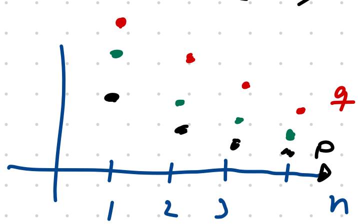
$$\alpha_n = \alpha + O(n^{-p})$$

Remark 2: If $0 < q < p$, and $\alpha_n = \alpha + O(n^{-p})$

then $\alpha_n = \alpha + O(n^{-q})$

e.g. $p = 5$, $q = 2$

$$|\alpha_n - \alpha| \le K \left|\frac{1}{n^5}\right| \le K \left|\frac{1}{n^2}\right|$$



Ex 1: Let $n \geq 1$

$$\alpha_n = \frac{n+1}{n^2} \quad (\alpha = 0)$$

$$|\alpha_n - 0| = \left|\frac{n+1}{n^2}\right| = \left|\frac{1}{n} + \underbrace{\frac{1}{n^2}}_{\leq \frac{1}{n}}\right| \leq \underbrace{2\left|\frac{1}{n}\right|}_{K|\beta_n|}$$

$$\Rightarrow \alpha_n = 0 + O\left(\frac{1}{n}\right)$$

$$\hat{\alpha}_n = \frac{n+1}{n^3} \quad (\alpha = 0)$$

$$|\hat{\alpha}_n - 0| = \left|\frac{n+1}{n^3}\right| = \left|\frac{1}{n^2} + \underbrace{\frac{1}{n^3}}_{\leq \frac{1}{n^2}}\right| \leq 2\left|\frac{1}{n^2}\right|$$

$$\Rightarrow \hat{\alpha}_n = 0 + O\left(\frac{1}{n^2}\right)$$ $\{\hat{\alpha}_n\}$ converges faster!

Similarly for functions:

Let $\lim_{h \to 0} F(h) = L$

Q: How fast is F approaching L? (as $h \to 0$)

Use known function $G(h)$, where $\lim_{h \to 0} G(h) = 0$

Def: Let $\lim_{h \to 0} F(h) = L$, $\lim_{h \to 0} G(h) = 0$. If there exists $k > 0$, $h_0 > 0$ s.t. $|F(h) - L| \leq K|G(h)|$ for $h \leq h_0$.

Then we write

$$F(h) = L + O(G(h))$$

Def: Let $\lim_{h\to 0} F(h) = L$, $\lim_{h\to 0} G(h) = 0$. If there exists

$K>0$, $h_0>0$ s.t. $|F(h)-L| \le K|G(h)|$ for $h \le h_0$.

Then we write
$$F(h) = L + O(G(h))$$

Remark: $G(h)$ is usually chosen as h^p $(p>0)$

and we're interested in
$$\max\{p : F(h) = L + O(h^p)\}$$

Ex 1: Analyze the conv. rate of
$F(h) = \sin(h) - h\cos(h)$ as $h \to 0$ $(L=0)$

Sol: Note by Taylor's Thm, that

$$\sin(h) = h - \frac{h^3}{6}\cos(\xi) \quad \text{where } 0 \le \xi \le h$$

$$\cos(h) = 1 - \frac{h^2}{2}\cos(\eta) \quad \text{where } 0 \le \eta \le h$$

$$\begin{aligned}
|\sin(h) - h\cos(h)| &= \left|\cancel{h} - \frac{h^3}{6}\cos(\xi) \cancel{-h} + \frac{h^3}{2}\cos(\eta)\right| \\
&\le \left|\frac{h^3}{6}\underbrace{\cos(\xi)}_{\le 1}\right| + \left|\frac{h^3}{2}\underbrace{\cos(\eta)}_{\le 1}\right| \\
&\le \left(\frac{1}{6} + \frac{1}{2}\right)|h^3|
\end{aligned}$$

$$\Rightarrow \underbrace{\sin(h) - h\cos(h)}_{F(h)} = 0 + O(h^3)$$