Math 471. Numerical methods
Introduction

Section 1.1–1.4 of Bradie

§1.1 Algorithms

Here is an analogy between Numerical Methods and Gastronomy:

Calculus, Lin Alg., Diff. eq.	$\longleftrightarrow$	Ingredients
Algorithm	$\longleftrightarrow$	Recipe
Coding	$\longleftrightarrow$	Cooking

Algorithms bridge abstract mathematical ideas and practical implementation of these ideas. They are often written in pseudo-code and require translation into computer codes. One should be comfortable with such convertion after several practices. What is mathematically more important, **however**, is the theoretical principles that drive these algorithms. This is the main task of this course. The name of "Numerical Analysis" is often used in lieu of "Numerical Methods" so as to emphasize the theoretical flavor of this subject.

• Example. Find the cubic root of a real number a using the iterative scheme

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n^2}\right) \tag{1}$$

The pseudo-code will look like

********** Algorithm 1 **********

Input	a
Initiate	x_1=*initial guess*
Loop	*while (stop criterion is not met)*
Use (1)	$x_{n+1} = 0.5 * (x_n + a/x_n/x_n)$
	end while loop
Output	x_n

********* end of Algorithm 1 **********

Questions to ask before coding:

1. Is this the right scheme? Translated into Math terms, can we prove the convergence

$$\lim_{n\to\infty} x_n = a^{1/3} \quad \text{for any} \quad a?$$

2. What initial guess should we use for x? Certainly not 0 or too close to 0. Will the initial guess affect the convergence?

3. How fast does x_n converge to a if it converges at all?

4. What is the stop criterion?

§1.2 Convergence

Convergence is a central tool for analyzing the performance of an algorithm. To quantify, one uses **convergence rate**: we call

$$\text{sequence } \{x_n\} \text{ converges to } x_\infty \text{ at rate } O(r_n)$$

if

$$|x_n - x_\infty| \le C r_n$$

for some fixed constant C and sufficiently large n. Here, we do require $\lim_{n\to\infty} r_n = 0$ so that, by the comparison method, the sequence of interest $\{x_n\}$ also converges, i.e. $\lim_{n\to\infty} |x_n - x_\infty| = 0$.

Remark: the phrase "for sufficiently large n" is understood as "for all $n \ge N$" with N being a fixed number.

Usually, the convergence rate r_n is a simpler expression than x_n itself. It gives us certain confidence that the error $x_n - x_\infty$ is at worst Cr_n. Also, the actual value of C is less important and is often omitted in discussions.

• Example. $\sin(1/n)$ converges to 0 at rate $O(1/n)$. Why? $|\sin(x)| \le |x|$ for real x.

• Example. Having the availability of the following identity (the proof of which is nontrivial already!)

$$\pi = 4\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}, \tag{2}$$

one can use the partial sum of the first $m+1$ terms,

$$S_m = 4\sum_{n=0}^{m} \frac{(-1)^n}{2n+1}$$

to approximate the numerical value of π. Obviously, the larger m is, the closer S_m is to π. But how sure are you about the accuracy of S_m? The answer is

$$S_m \text{ converges to } \pi \text{ at rate } O(1/m).$$

Why? We learned alternating series in Calculus and it turns out to be handy here. Write (2) out,

$$\pi = 4(1 - 1/3 + 1/5 - 1/7 + 1/9...)$$

which is an alternating series. The difference $\pi - S_m$, also called the truncation error, is

$$\pi - S_m = 4 \sum_{n=m+1}^{\infty} \frac{(-1)^n}{2n+1} = 4 \left(\frac{(-1)^{m+1}}{2m+3} + \frac{(-1)^{m+2}}{2m+5} + \frac{(-1)^{m+3}}{2m+7} ... \right)$$

So it boils down to show that the above error is bounded by a constant times $1/m$. To do this, we factor out $(-1)^m$,

$$\pi - S_m = 4(-1)^m \left(-\frac{1}{2m+3} + \frac{1}{2m+5} - \frac{1}{2m+7} ... \right).$$

Remember the trick of dealing with alternating series: pair up adjacent terms and they will give a definite sign; the leaving the first term alone and do the pairing again.

$$\pi - S_m = 4(-1)^m \left(\underbrace{-\frac{1}{2m+3} + \frac{1}{2m+5}}_{\text{negative}} \quad \underbrace{-\frac{1}{2m+7} + \frac{1}{2m+9}}_{\text{negative}} \quad ... \right)$$

which implies

$$\frac{\pi - S_m}{4(-1)^m} < 0.$$

Do another pairing but leave the first term out,

$$\pi - S_m = 4(-1)^m \left(-\tfrac{1}{2m+3} + \quad \underbrace{\frac{1}{2m+5} - \frac{1}{2m+7}}_{\text{postive}} \quad \underbrace{+\frac{1}{2m+9} - \frac{1}{2m+11}}_{\text{positive}} \quad ... \right)$$

which implies

$$\frac{\pi - S_m}{4(-1)^m} > -\frac{1}{2m+3}.$$

Now that we have lower bound AND upper bound for $\frac{\pi - S_m}{4(-1)^m}$, we estimate

$$\left| \frac{\pi - S_m}{4(-1)^m} \right| \leq \frac{1}{2m+3} < \frac{1}{m}$$

which, by the definition of convergence rate, leads to the desired conclusion.

Another quantification of convergence is **order of convergence**: we call $\{x_n\}$ converges to x_∞ of order α with asymptotic error constant λ, if

$$\lim_{n \to \infty} \frac{|x_{n+1} - x_\infty|}{|x_n - x_\infty|^\alpha} = \lambda.$$

Here, α, λ are both positive.

We will elaborate this concept later when going to fixed point iteration. To give a simple example, one can look at the geometric sequence $\{2^{-n}\}$. It converges to 0 at order $\alpha = 1$ and asymptotic error constant $\lambda = 1/2$.

$\alpha = 1$: linear convergence;

$\alpha = 2$: quadratic convergence;

$\alpha = 3$: cubic convergence; ....

§ 1.3 Floating point number system

β base. n number of significant digits. k exponent.

$$\pm(0.\overset{siginificand}{d_1 d_2 ... d_n})_\beta \times \beta^k, \qquad d_1 \neq 0 \text{ except for zero}$$

Given number $x.y$, how to calculate its floating point representation with base β? (optional)

Roundoff error is introduced by converting a real number to its floating point equivalent.

IEEE standard. e.g. double precision $\beta = 2$. Each number occupies 64 bits.

$$\overset{one\ number}{64bits} = \overset{two\ signs}{2bits} + \overset{siginificand}{52bits} + \overset{exponent}{10bits}$$

Resources of error: theory, model, algorithm (numerical methods), roundoff error, experimental data ...

• Example. An object with measurements $volume = 5 \pm 2m^2$ and $mass = 1 \pm 0.001kg$. Find the range of its density.

• Example. In a number system with $\beta = 10$ and $n = 4$, the numbers $a = 1$ and $b = 1.0001$ are considered "the same" due to roundoff. Then calculation $\dfrac{2}{a-b}$ becomes N/A. This is called **cancellation error** $\rightarrow$ subtraction of nearly equal numbers should be computed in other ways (see next section)

§ 1.4 Finite precision arithmetic

Finite precision of computer algorithm is due to the finite digits used in a floating point system.

A major disadvantage of floating point system is **cancellation error**. It happens in such calculation $b - a$ that $|b - a| \ll |b + a|$ (decrease of maginitude $\rightarrow$ loss of significant digits).

Quadratic formula.

$$ax^2 + bx + c = 0 \to x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The cancellation error from the numerator may cause inaccurate siginificand, esp. when one root is close to zero and the other is not (ill conditioned).

• Example. Consider $0.2x^2 - 47.91x + 6 = 0$. To 10 sinificant digits (higher precision), the roots are 0.2394246996×10^3 and 0.1253003555. But with 4 digits (lower precision), the arithmetic gives 0.2394×10^3 (ok) and 0.15 (loss of precision).

$$\text{note: } \sqrt{47.91^2 - 4 \cdot 0.2 \cdot 0.6} = \sqrt{2295 - 4.8} = \sqrt{2290} = 47.85$$

$$x_{1,2} = \frac{47.91 \pm 47.85}{0.4} \text{ cancellation error in the small root.}$$

To avoid subtraction of nearly equal numbers, reformulate the problem by *rationalizing* the numerator

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b + \sqrt{b^2 - 4ac}} = \frac{2 \cdot 6}{47.91 + 47.85} = 0.1253 \text{ ok!}$$

Finite difference. Heuristic: $f'(x) = \lim_{y \to x} \dfrac{f(y) - f(x)}{y - x}$. Thus,

$$f'(x) \approx D_+ f(x) = \frac{f(x + h) - f(x)}{h} \text{ for small } h > 0. \tag{3}$$

Called forward difference approximation. Also, backward difference, centered difference ...

The error comes from two sources:

1. truncation error due to discretization (derivative $\to$ finite difference). By the Taylor series $f(x + h) = f(x) + f'(x)h + f''(x)h^2/2 + ... = f(x) + f'(x)h + f''(\xi)h^2/2$ for some $\xi \in [x, x + h]$, we have

   $$D_+ f(x) = \frac{f(x + h) - f(x)}{h} = f'(x) + f''(\xi)h/2 = f'(x) + O(h).$$

   So the truncation error$\leq Ch$ where constant $C = \max f''(\xi)/2$.

2. roundoff error $\approx$ "machine precision $2^{-52} \times f(x)$" in computing $f(x + h) - f(x)$. Then, this error is amplified by a factor of $1/h$ in (3).

   $$\frac{2^{-52} \times f(x)}{h}$$

So, the total error is majorized by one of the two, depending on $h > \sqrt{2^{-52}}$ or $h < \sqrt{2^{-52}}$.

• Example. If $f(x) = e^x$ and $x = 1$, then $f'(1) = e = 2.71828...$ is the exact value. By setting, for instance, $h = 0.05, 0.025, 0.125$ and $h = 2^{-28}, 2^{-35}, 10^{-48}$, we calculate the following table

h	D_+f	$D_+f - f'(x)$	$(D_+f - f'(x))/h$
0.05	2.7874	0.0691	1.3821
0.025	2.7525	0.0343	1.3705
0.0125	2.7353	0.0171	1.3648
(As $h \to 0$, the tendency of convergence seems ...)			
"0"	e	"0"	e/2
(however)			
2^{-28}	2.718	3.666×10^{-8}	9.841
2^{-35}	2.718	1.041×10^{-5}	3.576×10^{5}
2^{-48}	**2.75**	**0.0317**	8.928×10^{12}

Each line in this table can be calculated using Matlab code

```
h=0.05;
a(1)=h; a(2)=(exp(1+h)-exp(1))/h; a(3)=a(2)-exp(1); a(4)=a(3)/h;
a
```

The last line, without semicolon, prints out the value of `a` in the command window.

To make the above code reusable, we encapsulate it in a function `Ch1F1`

```
function a=Ch1F1(h)
a(1)=h; a(2)=(exp(1+h)-exp(1))/h; a(3)=a(2)-exp(1); a(4)=a(3)/h;
```

Store this code in a file named `Ch1F1.m` and call this function from the command window or within another function.

Tip: use `format short e` before displaying an array so that each entry of the array has its own exponent. Try `format short` and see the difference.