AMSC460: Computational Methods

James Zhang[*]

August 29, 2024

These are my notes for UMD's CMSC460: Computational Methods. These notes are taken live in class ("live-TEX"-ed). This course is taught by Professor Haizhao Yang. The textbook for the course is *A First Course in Numerical Methods* by OU Ascher and Chen Greif.

Contents

[*]Email: jzhang72@terpmail.umd.edu

§1 Scientific Computing

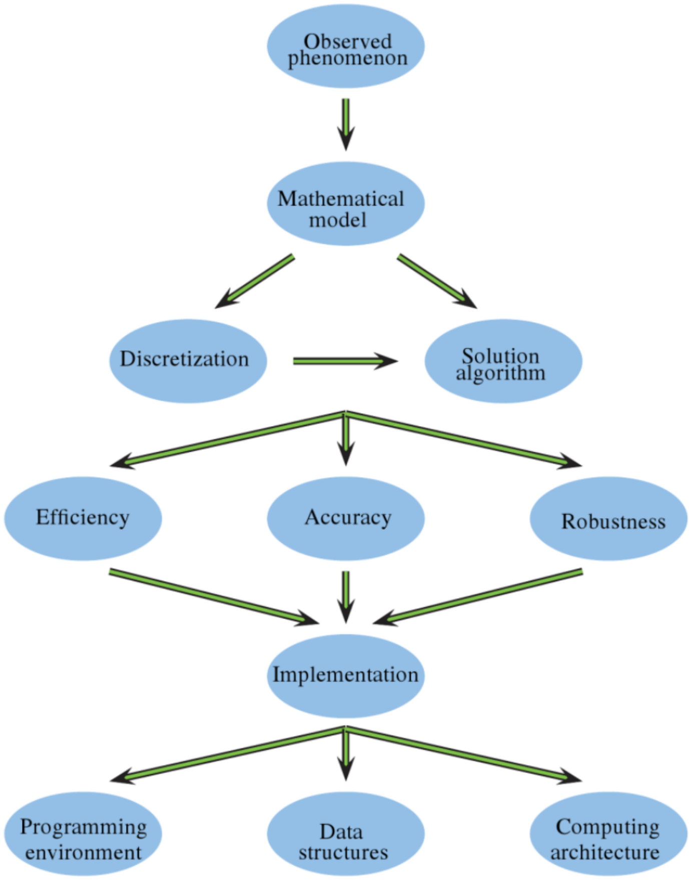

Figure 1.1. *Scientific computing.*

Definition 1.1 (Relative and Absolute Errors)**.** Let the target value be $u \in \mathbb{R}$ and let the numerical solution be $v \in \mathbb{R}$, then the **absolute error** is $|u - v|$ and the **relative error** is $\frac{|u-v|}{|u|}$.

Note 1.2. If $|u|$ is large, then relative error is used and if $|u|$ is very small, then relative error is not a good measurement.

Example 1.3 (The Stirling Approximation)

The formula $u = S_n = \sqrt{2\pi n} \cdot (\frac{n}{e})^n$ is used to approximate $n!$.

```
e = exp(1);
n = 1 : 10;
# Note that the following are vectors of len 10.
S_n = sqrt(2*pi*n).*((n/e).^n);
# Compute absolute and relative err.
fact_n = factorial(n);
abs_err = abs(fact_n - S_n);
rel_err = abs_err./abs(fact_n);
format short g
[n; fact_g; abs_err; real_err;]'
```

§1.1 Numerical Algorithms and Errors

Definition 1.4 (Error Types)**.**

1. Error in the problem to be solved. These may be errors in the mathematical model or errors in the input data.
2. Approximation errors, which can consist of discretization errors (errors in interpolation, differentiation, integration) or convergence errors, which can also arrive in iterative methods
3. Roundoff errors

Definition 1.5 (Taylor Series)**.** Assume that $f(x)$ has $k+1$ derivatives in an interval containing the point x_0 and $x_0 + k$. Then

$$f(x_0 + k) = f(x_0) + hf'(x_0) + \frac{h^2}{2} + \cdots + \frac{h^k}{k!} f^{(k)}(x_0) + \frac{h^{k+1}}{(k+1)!} f^{(k+1)}(\xi)$$

where ξ is some point between x_0 and $x_0 + h$, and the term $\frac{h^{k+1}}{(k+1)!} f^{(k+1)}(\xi)$ is the remainder term.

Note 1.6. To find $f'(x_0)$ observe that

$$f'(x_0) = \frac{f(x_0 + k) - f(x_0)}{h}$$

and then if we take the limit of this

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + k) - f(x_0)}{h}$$

we recover the h-definition of a derivative, and the discretization error is $\frac{h}{2} f''(\xi)$ because our model is

$$hf'(x_0) = f(x_0 + k) - f(x_0) - \frac{h^2}{2} f''(x_0) + \cdots$$

$$\implies f'(x_0) = \frac{f(x_0+k) - f(x_0)}{h} - \left(\frac{h}{2}f''(x_0) + \cdots\right)$$

$$\implies \left|f'(x_0) - \frac{f(x_0+k) - f(x_0)}{h}\right| = \left|\frac{h}{2}f''(x_0) + \cdots\right|$$

Definition 1.7 (Big-$\mathcal{O}$ and Θ Notation)**.** We define these for error characterization in terms of some parameters.

$$\begin{cases} h \text{ represents a small parameter} \\ n \text{ represents a large parameter} \end{cases}$$

Note 1.8. An error e depending on h we denote $e = \mathcal{O}(h^q)$ and if there are two positive constants q and C such that

$$|e| \leq Ch^q$$

Similarly, we write $e = \Theta(h^q)$ if $\exists\ C_1, C_2$ and $q > 0$ such that

$$C_1 h^q \leq |e| \leq C_2 h^q$$

n represents the problem size and then we use big $\mathcal{O}$ and Θ to denote the time or memory complexity of an algorithm.

Example 1.9

If we say $T = \Theta(n \log n)$ then we find C_1, C_2, x_0 such that

$$C_1 n \log n \leq T \leq C_2 n \log n \ \forall\ x \geq x_0$$

Note 1.10. Note that errors go down and then back up as h changes. A small number divided by another small number is a dangerous, think about exploding and vanishing gradients when training neural networks.

h	Absolute error
0.1	4.716676e-2
0.01	4.666196e-3
0.001	4.660799e-4
1.e-4	4.660256e-5
1.e-7	4.619326e-8

h	Absolute error
1.e-8	4.361050e-10
1.e-9	5.594726e-8
1.e-10	1.669696e-7
1.e-11	7.938531e-6
1.e-13	4.250484e-4
1.e-15	8.173146e-2
1.e-16	3.623578e-1

Note 1.11. In practice, error is the sum of **discretization error** + **rounding error**

§1.2 Algorithm Properties

Some good assessments of the quality of an algorithm are accuracy, efficiency, and robustness

Definition 1.12 (Accumulated Error)**.** Suppose you're evaluating polynomial with large degree. Your error e_1 from p_1 gets compounded by e_2 from p_2 and so on and so forth, so the total error follows

$$\text{Total error } \leq |e_1| + \cdots + |e_n|$$

§2 Binary Representation, Rounding Errors, Truncation Errors

Remark 2.1. Math claim: Any real number $x \in \mathbb{R}$ is accurately representable by an infinite sequence of digits, eg. $x \approx \pm 1$

$$x = \pm c(1.\{d_1\} \ldots \{d_{t+1}\} \cdots) \times 2^e$$

where $d_1, d_2, \cdots$ are integer numbers 0 or 1, and e is the integer exponent.
For each e, you can find a sequence to represent this real number x.

Example 2.2

Let $x = -(1.10110\cdots) \times 2^1$ which means $x = -1 + 1 \times \frac{1}{2} + 0 \times \frac{1}{2^2} + 1 \times \frac{1}{2^3} + \cdots$ where the $\frac{1}{2^t}$ term, we denote as $\frac{1}{s^t}$

Definition 2.3 (Truncating)**.** Chopping ignores digits $d_t, d_{t+1}, \cdots$ yielding $\widetilde{x} = \pm(1.\{d_1\} \cdots \{d_{t-1}\}) \times 2^e \approx x$ and so the error is $\mathcal{O}(\frac{1}{2^{te}}) = \mathcal{O}(\frac{1}{s^t})$

Definition 2.4 (Rounding)**.**