Topic and contents

School of Mathematics and Statistics

UNSW Sydney, Australia

MATH2089 – Numerical Methods

Week 01 – Computing with real numbers

Four steps in engineering problem analysis

Most engineering problems involve

- Problem specification/description, simplifying assumptions;
- Mathematical model: governing equations from physical laws
  - equilibrium equation
  - Newton's laws of motion
  - conservation of mass
  - conservation of energy
  - chemical balances
  - ...
- Solution techniques
  - analytical solutions
  - numerical solutions
- Interpretation of the solution
  - Visualisation
  - Physical model

Analytical vs numerical solutions

- Analytical solution is an exact answer
  - closed-form mathematical expression
  - involves the variables describing the problem being solved
- Very few practical problems have analytical solutions.
- Numerical solution
  - obtained by using a calculation-intensive process.
  - cannot be given as a mathematical expression
- Modern computers are essential
  - Efficient numerical methods: Time, Storage (memory, disk)
  - Modern computers: more memory, parallel and multi-core CPUs
- Engineering and Science have both changed
  - Physical models
  - Theoretical models
  - Computational models: computer simulation
- Understand the techniques, limitations

Computer architecture

Central Processing unit (CPU)

- consists of one or more cores
- running at a (variable) clock speed measured in GHz
  - 1 GHz $= 10^9$ cycles per second

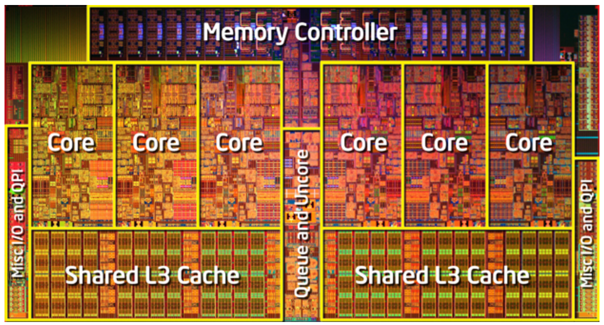


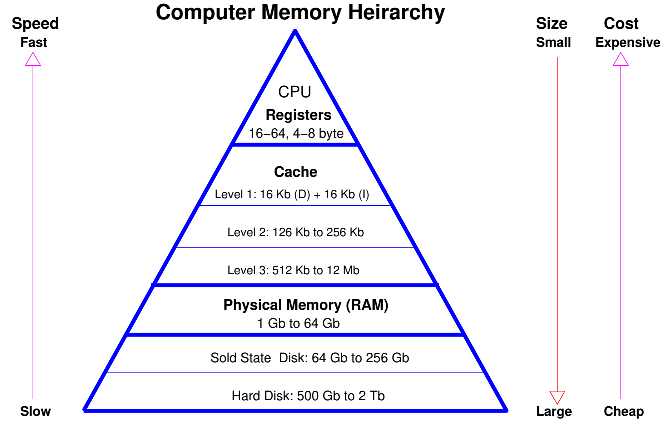
Figure: Intel core i7 980x die. 1.17 billion transistors, 240 mm^2

Binary computer – units

- Computers are binary: Store/work with bits
  - Bit, single binary digit: 1/0, on/off, true/false
- Combinations of bits give
  - Byte $= 8$ bits, an ASCII character, (2 bytes for extended characters)
  - 4 bytes $=$ 32 bits, integer or single precision floating point number
  - 8 bytes $=$ 64 bits, (long) integer or double precision floating point number
  - Kilobyte (Kb) $= 2^{10} = 1024 \approx 10^3$ bytes
  - Megabyte (Mb) $= 2^{10}$ Kb $= 2^{20} \approx 10^6$ bytes
  - Gigabyte (Gb) $= 2^{10}$ Mb $= 2^{20}$ Kb $= 2^{30} \approx 10^9$ bytes
  - Terabyte (Tb) $= 2^{10}$ Gb $= 2^{20}$ Mb $= 2^{30}$ Kb $= 2^{40} \approx 10^{12}$ bytes
  - Petabyte (Pb) $= 2^{10}$ Tb $= \cdots = 2^{40}$ Kb $= 2^{50} \approx 10^{15}$ bytes
  - ...
- Useful: $2^{10} = 1024 \approx 10^3$

Memory Hierarchy

- Volatile: CPU registers, cache, RAM
- Persistent: SSD, Hard disk, tape, CD, DVD



Numbers on a computer

- Computers use binary (base 2) storage/arithmetic
- Integers: sign, binary digits
  - Exact if integers are within limits
  - Largest signed 32 bit integer

$$2^{31} - 1 = 2147483647 \approx 2.1 \times 10^9$$

  - Largest signed 64 bit integer

$$2^{63} - 1 = 9223372036854775807 \approx 9.2 \times 10^{18}$$

  - MATLAB `intmax`
- Floating point numbers: approximate real numbers
  - sign, fraction (mantissa), exponent
  - Standard types
    - Single precision (32 bits, 4 bytes)
    - Double precision (64 bits, 8 bytes) (MATLAB)
  - not evenly spaced $\Longrightarrow$ round real to closest floating point number
- Tradeoff: storage vs accuracy

Storage

Example (Storing a matrix)

What is the largest n by n matrix that can be stored in 512 Kb cache, assuming each element requires 8 bytes?

Solution

Storage required for n by n matrix $= 8n^2$ bytes

$$8n^2 = 512 \times 2^{10} = 2^{19} \quad \Longrightarrow \quad n^2 = 2^{16} \quad \Longrightarrow \quad n = 2^8 = 256.$$

- n is size of a matrix, so it must be an integer
- largest possible (unachievable) value for n
- often want order of magnitude rather exact value

Memory limitations

Example (Memory limitations)

What is the largest amount of memory that can be addressed using an unsigned 32 bit pointer if each byte has an individual address (32 bit operating system)?

Solution

Maximum unsigned 32 bit integer is $2^{32} - 1$, so largest memory is

$$2^{32}\mathit{bytes} = 2^{22}\mathit{Kb} = 2^{12}\mathit{Mb} = 4\mathit{Gb}$$

- 32 bit operating system can use a maximum of 4 Gb of RAM
- Move to 64 bit operating systems (Windows or Linux)
- What is the largest amount of RAM for a 64 bit operating system?

Representing real numbers

- Decimal (base 10) representation

$$452.905 = (452.905)_{10}$$
$$= 4\times10^2 + 5\times10^1 + 2\times10^0 + 9\times10^{-1} + 0\times10^{-2} + 5\times10^{-3}$$

  - Non-normalized forms

$$0.452905\times10^3 = 0.0452905\times10^4 = 45.2905\times10^1$$

  - Normalized form (one nonzero digit before decimal point)

$$4.52905\times10^2$$

- Binary (base 2) representation: MATLAB `fp2bin.m`

$$9.90625 = (1001.11101)_2 = (1.00111101)_2 \times 2^{(11)_2} \quad \text{(normalized)}$$

$$\frac{1}{10} = (0.00011\,0011\,0011\,0011\cdots)_2$$

$1/10$ cannot be represented exactly on a binary computer

Floating point numbers I

Definition (Floating point numbers)

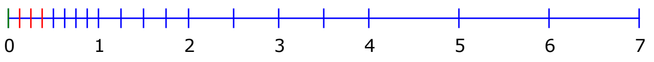
Floating point numbers are computer approximations to reals. They have the form

$$\pm(1.d_1 d_2 \dots d_p)_2 \times 2^e, \quad e_{\min} \le e \le e_{\max},$$

with $d_j = 0$ or 1, $j = 1, \dots, p$ for some $p > 0$.

Example (Simple floating point number systems)

Describe the floating-point number system with $p = 2$, $e_{\max} = 2$ and $e_{\min} = -1$. MATLAB `floatgui.m`



Floating point numbers II

- IEEE 754 standard `http://grouper.ieee.org/groups/754/`

Precision	Bits used	p	$e_{\max}$	$e_{\min}$
Single	32	23	127	-126
Double	64	52	1023	-1022

- Characteristics of floating point numbers
  - Relative machine precision: smallest $\epsilon > 0$ such that $1 + \epsilon > 1$ on the computer (MATLAB `eps`)
  - Largest floating point number (MATLAB `realmax`)
  - Smallest positive floating point number (MATLAB `realmin`)
  - Number of significant figures (nsf)
- Values

Precision	`eps`	`realmax`	`realmin`	nsf
Single	1.2×10^{-7}	3.4×10^{38}	1.2×10^{-38}	7
Double	2.2×10^{-16}	1.8×10^{308}	2.2×10^{-308}	15

IEEE extensions – `Inf`, `NaN`

Definition (Infinity and Not a Number)

- Overflow: Any number larger than the largest possible floating point number is represented by `Inf`
- Any number less that the most negative floating point number is represented by `-Inf`
- Not a Number: Any result that is mathematically not defined is represented by `NaN`

- `Inf` behaves like ∞, propagates, but not the same
  - Evaluate 1.8×10^{308}, -1.8×10^{308} (MATLAB or Excel)
  - Evaluate $2.4/0$, $-3.6/0$
  - Evaluate $\pi + \texttt{Inf}$, $3.6 * \texttt{Inf}$
- `NaN` propagates
  - Evaluate $0/0$, $\texttt{Inf} - \texttt{Inf}$, $\texttt{Inf}/\texttt{Inf}$
  - Evaluate $9.3 * \texttt{NaN}$, $\sin(\texttt{NaN})$

Underflow

Definition (Underflow)

A number whose magnitude is less than the smallest positive floating point number underflows to 0

Example (Gradual underflow)

1. Find x such that $e^{-x} = \texttt{realmin}$
2. Calculate $f(x) = e^{-x}$ for $x = 700, 701, ..., 750$

Absolute and relative errors

Definition (Absolute and relative error)

Let $\bar{x}$ be a (computed) approximation to x.

- The absolute error is $\text{ae}(\bar{x}) = |\bar{x} - x|$
- For $x \neq 0$ the relative error is $\text{re}(\bar{x}) = \dfrac{|\bar{x} - x|}{|x|}$

Example (Absolute and relative error)

1. Calculate the absolute and relative error in using the following approximations to π.
   - $a = 3$
   - $b = 22/7$
   - $c = 355/113$

Rounding error

Definition (Rounding error)

The rounding error in storing x on a computer is the absolute error between the exact value of x and the value stored on the computer.

Estimating rounding error

- ϵ is the relative machine precision on a computer
- $\epsilon|x|$ gives an estimate of the rounding error (absolute error) in storing x on a computer.

Example (Rounding error)

Estimate the rounding errors in storing the following values:

1. $x = 0.12$.
2. $y = 1.2$ billion.

Practical considerations

Definition (Significant figures)

An approximation $\bar{x}$ to a true value $x \neq 0$ is correct to p significant figures if p is the largest positive integer such that

$$\left|\frac{x - \bar{x}}{x}\right| < \frac{1}{2}\,10^{-p}$$

- x has a relative error less than 0.5×10^{-p}
  $\implies x$ has at least p correct significant digits
- Never give an answer to more accuracy than your input data
- $\bar{x}$ is correct to p decimal places if the absolute error less than 0.5×10^{-p}.

Catastrophic cancellation

Definition (Catastrophic cancellation)

If $\bar{x}$ and $\bar{y}$ are (computed) approximations to non-zero quantities x and y where $x \approx y$, then

$$\text{relative error}(\bar{x} - \bar{y}) \gg \text{relative error}(\bar{x}) + \text{relative error}(\bar{y})$$

- Subtracting nearly equal quantities is dangerous

Example (Catastrophic cancellation)

Suppose $b = 998.216$ and $c = 998.251$ are measured to 6 significant figures.

1. Find the relative and absolute errors in b and c
2. Estimate the relative error in $b - c$.

Example

Example

Suppose $c \approx 101.25$ is measured to 2 decimal places. Estimate the absolute and relative errors in c. Estimate the rounding error when storing 101.25 on a computer in double precision.

Roots of a quadratic

Example (Roots of a quadratic)

The roots of the quadratic $ax^2 + bx + c$ are

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- When is this formula susceptible to catastrophic cancellation?
- Find a mathematically equivalent formula that removes this difficulty.

Solution (Formula for roots of a quadratic)

Floating point operation – flops

Definition (flops)

A floating point operation or flop is one of the basic arithmetic operations +, −, ∗, / applied to floating point numbers.

- flops or floating point operations per second is used to estimate the time an operation will take
- No distinction between these operations: in reality ∗, / considerably longer than +, −
- Units
  - Mflops, megaflops $= 10^6$ flops
  - Gflops, gigaflops $= 10^9$ flops
  - Tflops, teraflops $= 10^{12}$ flops
  - Pflops, petaflops $= 10^{15}$ flops

Estimating computation time

Estimate the time required for a computational task or the largest problem that can be solved in a given time, using

- flops required by computational task
- performance characteristics of available computer

Example (Matrix multiplication)

Suppose a 3 GHz dual core PC can do 2 flops per clock cycle. Matrix multiplication $C = AB$ where $A, B, C \in \mathbb{R}^{n\times n}$ is defined by

$$C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj} \qquad i = 1, \dots, n, \quad j = 1, \dots, n$$

1. How many flops are required to calculate C
2. Estimate how long it will take to calculate C for $n = 1000$.
3. Estimate the largest matrices that can be multiplied in 1 hour.

Matrix multiplication cont.

Solution (Matrix multiplication – computation time)

1. *Flops for matrix multiplication*
   - *Matrix multiplication requires n multiplications and n additions for each element C_{ij}.*
   - *Total of n^2 elements $C_{ij} \implies n^3$ multiplications and n^3 additions.*
   - *Total of $2n^3$ flops.*
2. *Time for $n = 1000$*
   - *2 flops per clock cycle, 3 GHz (cycles/second) $\implies 6 \times 10^9$ flops/sec.*
   - *$n = 10^3 \implies$ matrix multiplication requires $2n^3 = 2 \times 10^9$ flops*
   - *Time = (total flops) / (flops/sec) $= (2 \times 10^9)/(6 \times 10^9) = 1/3$ sec*
3. *Largest matrix multiplication in 1 hour*
   - *1 hour $= 60 \times 60 = 3600$ seconds*
   - *1 hour $\implies 3600 \times 6 \times 10^9 = 2.16 \times 10^{13}$ flops*
   - *Solve $2n^3 = 2.16 \times 10^{13} \implies n = (1.08 \times 10^{13})^{\frac{1}{3}} \approx 2.2 \times 10^4$.*
   - *n is integer around $22,000$.*

Example

Example

Suppose you are working with a 2.5 GHz dual core PC that can do 2 flops per core per clock cycle. Given an input size of n, suppose that some algorithm requires n^2 flops to compute a desired quantity.

- Estimate how long it would take the algorithm to compute the desired quantity for $n = 20,000$.
- Estimate the largest input size n for which the algorithm can compute the desired quantity in one minute.

Patriot missile disaster

Gulf war: Patriot missiles designed to shoot down incoming scud missiles

- Decimal vs binary

$$\begin{aligned}\frac{1}{10} &= \frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^8}+\frac{1}{2^9}+\frac{1}{2^{12}}+\frac{1}{2^{13}}+\cdots \\ &= 0.0001100110011001100110011001100....\end{aligned}$$

- 24 bit timer stored $\frac{1}{10}$ as $0.00011001100110011001100$
- Error $= 0.000000000000000000000000011001100\ldots$ binary $\approx 9.5\ 10^{-8}$
- Timer running 100 hours $\Longrightarrow$ error $= 9.5\ 10^{-8} \times 100 \times 60 \times 60 \times 10 = 0.34$ secs
- Scud $1,676$ metres/sec $\Longrightarrow$ error > 0.5 km
- Outside range gate of patriot tracking $\Longrightarrow$ 28 dead

References

- IEEE 754: Standard for Binary Floating Point arithmetic, http://grouper.ieee.org/groups/754/
- David Goldberg, What every Computer Scientist should know about floating point arithmetic, http://www.validlab.com/goldberg/paper.ps
- William Kahan, http://www.cs.berkeley.edu/~wkahan and http://www.cs.berkeley.edu/~wkahan/ieee754status/ieee754.ps
- Thomas Huckle, Collection of Software Bugs, http://www5.in.tum.de/~huckle/bugse.html
- The Risks Digest, http://catless.ncl.ac.uk/Risks/
- Wikipedia, Software Bug, http://en.wikipedia.org/wiki/Computer_bug
- http://www.youtube.com/watch?v=EMVBLg2MrLs