# UNIVERSITY OF NEW SOUTH WALES
# School of Mathematics and Statistics

# MATH2089 Numerical Methods and Statistics
# Term 2, 2019

## Numerical Methods Tutorial – Week 2 Solutions

1. The speed of light is $2.99792 \times 10^8$ m/sec in a vacuum. Estimate how far an electromagnetic signal in a circuit can move in 1 ns. What are the consequences of this?

   **Answer** One nanosecond or 1 ns is $10^{-9}$ of a second. Thus in 1 ns a signal moving at the speed of light would travel
   $$3 \times 10^8 \times 10^{-9} = 0.3 \text{ metres}.$$
   As an electromagnetic signal moves at a speed slightly less than the speed of light, this is an upper bound on how far an electromagnetic signal (in a computer) can move in 1 ns. This is one of the factors driving chip manufacturers to make smaller and smaller circuits. Another major limiting factor is the amount of heat produced by the circuits in the CPU.

2. (a) Calculate the absolute and relative errors when using 1.414 as an approximation to $\sqrt{2}$. Why are the absolute and relative errors similar in this example?

   (b) The volume $V$ of a tank is $1,034.46$ litres measured to two decimal places. Calculate the absolute and relative errors in $V$.

   (c) An instrument measures the temperature $T$ to 4 significant figures. Calculate the relative and absolute error when $T \approx 200$.

   (d) Let $f(x) = \sin(x) - x$. The relative error in $x$ and $\sin(x)$ as stored on a computer is given by the relative machine precision $\epsilon$. When using MATLAB, which uses double precision arithmetic,

      i. Calculate the absolute error in $x$ when $x = 0.1$.

      ii. If the absolute error in $f(x)$ is the same as the absolute error in $x$, estimate the relative error, and hence the number of significant figures, in $f(x)$.

   **Answer**

   (a) When using 1.414 as an approximation to $x = \sqrt{2}$,
   $$\begin{aligned} \text{absoluter error} &= |\sqrt{2} - 1.414| \approx 2.1 \times 10^{-4} \\ \text{relative error} &= \frac{\text{absolute error}}{|x|} = \frac{|\sqrt{2} - 1.414|}{\sqrt{2}} \approx 1.5 \times 10^{-4} \end{aligned}$$
   In this case the absolute and relative errors are similar as $|x| \approx 1$.

   (b) As $V$ has been measured to two decimal places the **absolute** error is less than $0.5 \times 10^{-2}$. The relative error is thus less than $0.5 \times 10^{-2}/|V| \approx 4.8 \times 10^{-6}$.

   (c) If $T$ has 4 significant figures then the **relative** error is less than $0.5 \times 10^{-4}$. The absolute error is then $|T| \times 0.5 \times 10^{-4} \approx 10^{-2}$.

   (d) i. The absolute error in storing $x$ on a computer using double precision is approximately $|x|\epsilon \approx 0.1 \times 2.2 \times 10^{-16} = 2.2 \times 10^{-17}$.

      ii. Suppose the absolute error in $f(x) = \sin(x) - x$ is also approximately $2.2 \times 10^{-17}$. As $f(0.1) \approx -1.6658 \times 10^{-4}$, the relative error in $f(0.1)$ is approximately $1.3 \times 10^{-13} = 0.13 \times 10^{-12}$. Thus we expect at least 12 significant figures in the computed value of $f(0.1)$.

3. You are working on a 3 GHz dual core computer that can do one flop per core per clock cycle.

   (a) What is the size $n$ of the largest $n$ by $n$ matrices that can be multiplied in 1 hour? Multiplying two $n$ by $n$ matrices requires $2n^3$ flops.

   (b) Assuming each element of a matrix is stored in double precision (8 bytes), how much memory will each of these matrices require?

   **Answer**

   (a) The speed of this computer is

   $$3 \times 10^9 \times 2 = 6 \times 10^9 \text{ flops per sec.}$$

   As we want the task done in one hour, we want the value of $n$ such that

   $$2n^3 = 60 \times 60 \times 6 \times 10^9 \Longrightarrow n^3 = 1.08 \times 10^{13} \Longrightarrow n \approx 22,000.$$

   (b) An $n$ by $n$ matrix has $n^2$ elements. Assuming each element is stored as a double precision floating point number taking 8 bytes, the total storage is

   $$8n^2 \text{ bytes} = 8(2.2 \times 10^4)^2 = 3.87 \times 10^9 \text{ bytes} = 3.87 \text{ GBytes}$$

   using the approximation 1 GByte $= 2^{30} \approx 10^9$ bytes.
   As most PCs have 4 GB or less of memory, and $C = AB$ requires 3 such matrices, it will be a lack of memory that prevents this computation being done rather than the computation time.

4. The number of flops required to solve an $n$ by $n$ linear system $A\mathbf{x} = \mathbf{b}$ by Gaussian elimination (row operations to reduce the system to Row-Echelon form, followed by back-substitution) is $\frac{2n^3}{3} + O(n^2)$. On a 3 GHz PC which can do one flop per clock cycle:

   (a) Estimate how long will it take to solve a linear system of size $n = 1000$.

   (b) Estimate the largest linear systems than can be solved in

      i. 1 minute,
      ii. 1 hour,
      iii. 1 day.

   **Answer** A 3 GHz PC doing one flop (floating point operation) per clock cycle, does $3 \times 10^9$ flops per second. Thus, considering only the highest order term,

   $$\text{Time} = \frac{\text{Number of flops required}}{\text{Number of flops per second}} = \frac{\frac{2}{3}n^3}{3 \times 10^9} \quad \text{secs} \tag{1}$$

   **Remark:** The definition of "Big O" is in the limit as $n \to \infty$, and in this limit any constant times $n^2$ is insignificant compared to $\frac{2n^3}{3}$. However if the constant in the $O(n^2)$ is huge (for example $10^9 n^2$) then $n$ will have to be very large before it is insignificant compared to $\frac{2n^3}{3}$. For example, how large must $n$ be before $10^9 n^2$ is less than 10% of $\frac{2n^3}{3}$?

   (a) For $n = 1000$,

   $$\text{Time} = \frac{\frac{2}{3}\left(10^3\right)^3}{3 \times 10^9} = \frac{2}{9} \approx 0.2 \text{ secs}$$

   (b) To find the largest system than can be solved in $T$ seconds you need to solve (1) for $n$,

   $$T = \frac{2n^3}{9 \times 10^9} \Longrightarrow n^3 = 4.5T \times 10^9 \Longrightarrow n = \left(4.5T \times 10^9\right)^{\frac{1}{3}}$$

i. $T = 1$ minute $\Longrightarrow T = 60$ seconds $\Longrightarrow n \approx 6,500$

ii. $T = 1$ hour $\Longrightarrow T = 60 \times 60$ seconds $\Longrightarrow n \approx 25,000$

iii. $T = 1$ day $\Longrightarrow T = 24 \times 60 \times 60$ seconds $\Longrightarrow n \approx 73,000$

The size $n$ must be an integer. As $O(n^2)$ terms have been ignored, only the first couple of digits are significant. For example $T = 1$ day gives $n = 72986.4$, but at most two of these digits are significant.

5. The Fast Fourier Transform (FFT), one of the great algorithms of the 20th century, is used to analyse signals (your mobile phone for example). The FFT takes $O(n\log_2(n))$ flops to process $n$ data values, rather than $O(n^2)$ flops for the straightforward Discrete Fourier Transform (DFT). Suppose you have a 2.5GHz quad core computer that can do 4 floating point operations per core per clock cycle.

   (a) Estimate how long will it take to process $n = 2^{30}$ data values using the DFT and the FFT.

   (b) Estimate the largest number $n$ of data values that can be processed in one second by the DFT and the FFT.

   **Answer** A 2.5 Ghz quad core computer that can do 4 floating point operations per core per clock cycle, can do

$$2.5 \times 10^9 \times 4 \times 4 = 4 \times 10^{10} \quad \text{flops/sec.}$$

   Note that for both the DFT and FFT the number of floating point operations are only given in terms of "Big O", so the constant that multiplies the terms is **not** known. We will assume it is just 1, and note that the answers are only approximate!

   (a) Consider a set of data with $n = 2^{30} \approx 10^9$.
   An implementation of the DFT requiring $O(n^2)$ flops will take

$$\frac{\text{number of flops required}}{\text{flops per sec}} = \frac{n^2}{4 \times 10^{10}} = \frac{\left(2^{30}\right)^2}{4 \times 10^{10}} \approx \frac{10^{18}}{4 \times 10^{10}} \approx 2.5 \times 10^7 \text{ secs} \approx 0.8 \text{ years.}$$

   An implementation of the FFT requiring $O(n\log_2(n))$ flops will take

$$\frac{\text{number of flops required}}{\text{flops per sec}} = \frac{n\log_2(n)}{4 \times 10^{10}} = \frac{2^{30} \times 30}{4 \times 10^{10}} \approx \frac{3 \times 10^{10}}{4 \times 10^{10}} = 0.75 \text{ secs.}$$

   Which algorithm are you going to use?

   (b) In one second the computer can do $4 \times 10^{10}$ floating point operations.
   The largest problem that can be done using a DFT taking $O(n^2)$ flops satisfies

$$n^2 = 4 \times 10^{10} \Longrightarrow n = 2 \times 10^5.$$

   The largest problem that can be done using a FFT taking $O(n\log_2(n))$ flops satisfies

$$n\log_2(n) = 4 \times 10^{10} \Longrightarrow n \approx 1.3 \times 10^9.$$

   This equation requires numerical solution (Week 4), but MATLAB code for it is

```
func = @(n) n.*log2(n) - 4e10;
n = fzero(func, 1e9)
```

   This uses the initial guess of $n = 10^9$ which takes 0.75 seconds from part a).