LECTURE 1 – INTRODUCTION

- **What is numerical analysis?**
  Development and analysis of numerical algorithms for scientific computing.
- **What is scientific computing?**
  Computational simulation of mathematical models of physical phenomena and the solution of scientific problems.

Examples

Here are a few examples shamelessly grabbed from YouTube:

- Mantle convection with RBFs:
  `http://www.youtube.com/watch?v=-kDb0HlDsIM`
- CFD wind simulation:
  `http://www.youtube.com/watch?v=0qRUtRytanI&t=60s`
- Tsunami modelling:
  `http://www.youtube.com/watch?v=2oPKyiA1hew`
- ...

Aims of the course

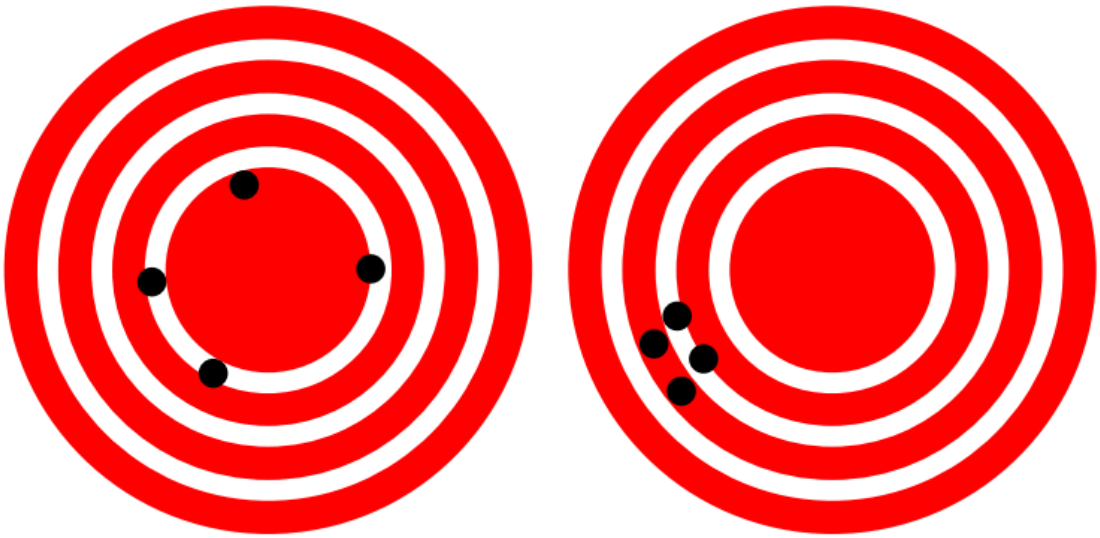
Aims:

- Learn what kind of problems are faced in scientific computing.
- Develop an understanding for what tools (i.e., algorithms) are available for tackling these problems.
- Gain experience in implementing algorithms and solving problems.

FINITE PRECISION ARITHMETIC

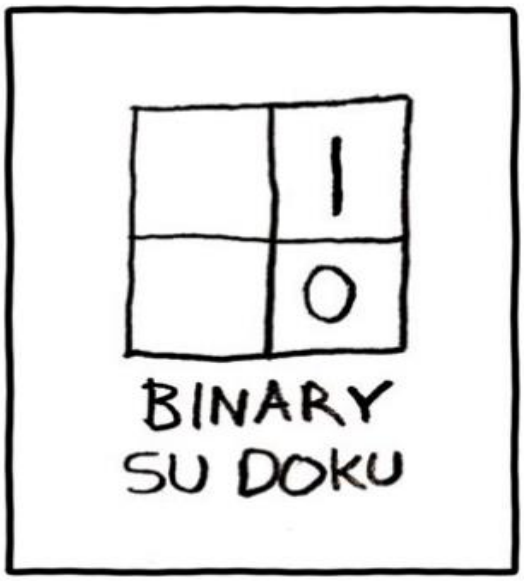
- Floating point representation
- IEE754 standard
- Floating point operations (FLOPS)
- Ill-conditioning / stability / error analysis

Accuracy vs Precision



- Accuracy and precision are often confused, but they are not the same thing
- Accuracy: the absolute or relative error of an approximate quantity
- Precision: the accuracy with which the basic arithmetic operations are performed
- Can be very different (e.g. in solution of ill-conditioned linear system)

Binary numbers



- How are numbers represented on a computer?
  Binary
- Example:

```
42 =     0 0 1 0 1 0 1 0
42.75 = 1 0 1 0 1 0.1 1
```

- Fixed point format → limited range:
  - Maximum value is small
  - Minimum value is large
- This is why we use floating point.

Floating point system

- Floating point format: `- - m - -.- - n - -` $\times 2^e$, $\quad e \in [e_{\min}, e_{\max}]$
- Exponent can lead to ambiguity. For example:
  - `4 = 0 1 0 0.0 0 0 0` $\times 2^0$
  - `4 = 0 0 1 0.0 0 0 0` $\times 2^1$
  - `4 = 0 0 0 1.0 0 0 0` $\times 2^2$
  - `4 = 1 0 0 0.0 0 0 0` $\times 2^{-1}$
- Ambiguity can be removed by always placing the point after the first 1, i.e.,
  - `4 = 1.0 0 0 0 0 0 0` $\times 2^2$ $\quad \leftarrow$ "normalized representation"
- This convention means we don't need to store the first bit! (It is implicit.)
  - `# = 1.- - - m - - -` $\times 2^e$ $\quad \leftarrow$ (More precision, a little more computation)

- `# = 1.- - - m - - -` $\times 2^e$
- Compare to scientific notation:
  - Revolution of Io (moon of Jupiter) $= 152853.5047\text{s}$
    $= 1.528535047 \times 10^5\text{s}$
- Implicitly storing the first 1 has at least one problem ...
- How to represent zero?!
  - This requires a convention (significand of 0, exponent of 0).
  - This effectively loses one bit of exponent.

- Floating point was adopted fairly early in computing, but not standardised.
- Standardisation arrived in 1985 with the adoption of the IEEE754 standard.
- The main architect was William Kahan (UC Berkeley).

```
-|-----------|-----------
s     e           m
```

Double precision:

- s = sign: 1 bit, 0 = +ve, 1 = -ve
- e = exponent: 11 bits, interpreted as $e - 1023$ (to avoid another sign bit).
  - $e_{\min} = 0 - 1023 + 1 = -1022$
  - $e_{\max} = (2^{11} - 1) - 1023 - 1 = 1023$
- m = significand: 52 bits (+ 1 implicit). Note $2^{-53} \approx 10^{-16}$.
- $\# = (-1)^s \left(1 + \sum_{j=1}^{52} m_j 2^{-j}\right) \times 2^{e-1023}$
- What are `realmin` and `realmax`? 2^{-1022} and $(2 - 2^{-52}) \times 2^{1023}$

Matlab Demo

```
>> [s, e, m] = decode_ieee(d)
```