Numerical Analysis Intro

How is your numerical computing intuition? Assume each of the following claims is referring to a computation being done on a computer with binary number representations (more on that later). Each may be true, false, or somewhere in-between. Discuss!

1. Every rational number can be stored exactly.
2. The distance between any two adjacent numbers is the same and is called machine epsilon.
3. All computed answers are wrong.
4. Accurate solutions are more important than fast solutions.
5. It is more difficult to find a solution of a problem when the solution is 0 than when the solution is 100.
6. Let $\mathbf{b}$ be a real vector size $n \times 1$ and A be a real $n \times n$ matrix. As long as A is invertible, the system $A\mathbf{x} = \mathbf{b}$ is solvable quickly and accurately.
7. Solving a problem 100 times (like $A\mathbf{x} = \mathbf{b}$ for the same A, different $\mathbf{b}$) takes 100 times as much work as solving the same problem once.
8. Storing every $m \times m$ matrix A requires storing m^2 numbers.
9. A calculator computes $\sin(\frac{\pi}{124})$ by looking it up in a table in memory.

Binary Number System

Developing an understanding of numerical techniques must begin with an understanding of the numbers on which the techniques operate. It is not difficult to imagine that storing an irrational number exactly is impossible on a computer - at some point we must stop storing additional digits. This is a source of error called *truncation error.* But in fact, the finite storage space per number affects far more than just irrational numbers, which we now explore.

Modern arithmetic models are based on binary representations of numbers. A base 10 number, such as $(321)_{10}$, means $3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$. In *binary* or *base 2*, the powers of 10's are replaced with powers of 2. So $(101)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (5)_{10}$. Binary digits are called *bits*, and are always 0 or 1.

Example 1. Convert each of the following binary numbers to their decimal representations.

- $(10110)_2$
- $(1100.01)_2$
- $(.111)_2$

Converting decimal to binary is a bit more complicated. For the integer part, repeatedly divide by 2, looking for the remainder; for the fractional part, repeatedly multiply by 2, and check if the result is more or less than 1.

Example 2. Convert $(11.3125)_{10}$ to binary.

We start with the integer part:

$$\begin{aligned} 11 \div 2 &= 5R1 \to 1 \\ 5 \div 2 &= 2R1 \to 1 \\ 2 \div 2 &= 1R0 \to 0 \\ 1 \div 2 &= 0R1 \to 1 \end{aligned}$$

So $(11)_{10} = (1011)_2$. Now the fractional part.

$$\begin{aligned} .3125 \cdot 2 &= 0 + .625 \to 0 \\ .625 \cdot 2 &= 1 + .25 \to 1 \\ .25 \cdot 2 &= 0 + .5 \to 0 \\ .5 \cdot 2 &= 1 \to 1 \end{aligned}$$

Thus $(.3125)_{10} = (.0101)_2$, and combining gives $(11.3125)_{10} = (1011.0101)_2$.

Example 3. Convert each of the following to binary representation.

- 0.5625
- 3.125
- $\frac{7}{10}$