

MA2252 Introduction to Computing

Lecture 9

Representation of Numbers

Sharad Kumar Keshari

School of Computing and Mathematical Sciences

University of Leicester

At the end of lecture, students will be able to understand

- Base-N numbers
- Binary numbers
- Floating Point Numbers

Motivation

Sometimes, MATLAB gives unexpected outputs. Consider these examples:

- $\exp(i\pi)+1==0$
- $\sin(\pi)$, $\cos(\pi/2)$
- $5+8e-17>5$
- $1 - 3*(4/3 - 1)$
- $2^{-1073}==0$, $2^{-1075}==0$
- $\text{sqrt}(1e-16 + 1) - 1$

The goal of this lecture is to understand the reason behind these strange results.

Base-N numbers

Base: Number of unique digits to represent a number.

Example:

Decimal number (Base-10 number)

- ① Uses digits from 0 to 9. So, base is 10.
- ② Decimal number can be expanded in powers of 10.

Examples:

$$582 = 5 \times 10^2 + 8 \times 10 + 2 \times 10^0,$$

$$25.36 = 2 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 6 \times 10^{-2}.$$

Other examples:

- **Binary number:** Uses only two digits: 0 and 1.
- **Octal number:** Uses digits from 0 to 7.
- **Hexadecimal number:** Uses 16 digits: 0-9 followed by A-F or a-f.

Binary numbers

- Binary numbers are represented by only 0s and 1s. So, the base is 2.
- A digit in binary number is called a **bit**.
- **Example:** The number 1001000110.

This number can be converted to a decimal number as follows:

$$1 \times 2^9 + 0 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 582$$

Binary numbers (contd.)

Pros

- Easier for a computer to store and work with.
- A computer can do all arithmetic operations with them.
- A 32-bit computer can represent upto 2^{23} binary numbers.

Cons

- Harder for humans to do binary algebra.
- Limited **range** and **precision** to perform all mathematical calculations.

Floating Point Numbers

MATLAB uses **floating point** numbers or **floats** to achieve the desired range and precision required to perform mathematical calculations.

Note: Floating point numbers are always **rational**. MATLAB approximates irrational numbers (e.g. π) by rational numbers.

Types of float:

- single precision (32 bits)
- double precision (64 bits)

Single precision float

In the **IEEE754** standard for single precision, a float which is represented as

$$n = -1^s 2^{e-127} (1 + f)$$

Here, s , e and f are sign indicator, exponent and fraction respectively.

32 bits are allocated as follows:

- s has 1 bit so it takes values of 0 or 1.
- e has 8 bits so it can take 2^8 values.
- f has 23 bits so it can take 2^{23} values. $0 \leq f < 1$.

Floating Point Numbers (contd.)

Example:

Convert the number

1 10000011 1100000000000000000000 (IEEE754)

into decimal (base 10) number.

Floating Point Numbers (contd.)

Solution:

$$\begin{aligned}s &= 1, \\ e &= 1 \times 2^7 + 0 + 0 + 0 + 0 + 0 + 1 \times 2^1 + 1 \times 2^0 = 131, \\ f &= 1 \times 2^{-1} + 1 \times 2^{-2} + \dots = 0.75.\end{aligned}$$

$$n = -1^1 2^{131-127} (1 + 0.75) = -2^4 \times 1.75 = -28.$$

Gaps between numbers

Not every decimal (base 10) number can be represented by double precision float. This causes gaps between numbers in MATLAB.

To find the gap, use MATLAB's `eps` function. `eps(x)` gives the gap between number `x` and next representable number.

Example:

`eps(5)=8.881784197001252e-16.`

The gap increases as numbers get large because the factor 2^{e-127} grows in size.

Special cases

① $e=0$ (00000000(base 2))

In this case, the float is calculated using

$$n = -1^s 2^{-126} f$$

② $e=255$ (11111111(base 2))

- $f \neq 0$: $n=\text{NaN}$ (Not a Number)
- $f=0$ and $s=0$: $n=\text{Inf}$
- $f=0$ and $s=1$: $n=-\text{Inf}$

Floating Point Numbers (contd.)

Single precision float interesting facts

- Largest defined number

0 11111110 111111111111111111111111
(3.402823466385289e+38)

- Smallest defined positive 'normal' number

0 00000001 000000000000000000000000
(1.175494350822288e-38)
(2^{-126})

Note: Use MATLAB functions `realmax('single')` and `realmin('single')` to find the above results.

Floating Point Numbers (contd.)

- Smallest defined subnormal number
0 00000000 000000000000000000000001
(1.401298464324817e-45)
(2^{-149})

Note: Use MATLAB `single()` function to represent a number in single precision.

Double precision float

MATLAB uses default double precision of **IEEE754** standard. A double precision is represented as

$$n = -1^s 2^{e-1023} (1 + f)$$

Again, s , e and f are sign indicator, exponent and fraction respectively.

64 bits are allocated as follows:

- s has 1 bit so it takes values of 0 or 1.
- e has 11 bits so it can take 2^{11} values.
- f has 52 bits so it can take 2^{52} values. $0 \leq f < 1$.

Floating Point Numbers (contd.)

The special cases for double precision float are similar to single-precision float.

Double precision float interesting facts

- Largest defined number
1.797693134862316e+308 (type `realmax` in command window)
- Smallest defined positive 'normal' number
2.225073858507201e-308 (type `realmin` in command window)
- Smallest defined subnormal number
4.940656458412465e-324
(2^{-1074})

Typing 1e309 in the command window gives the output

- ① 1.0000e+309
- ② Inf
- ③ NaN
- ④ Some error

To answer, please go to mentimeter link in the chat.

Overflow and Underflow

- Any number larger than the largest defined floating point number causes overflow. MATLAB assigns the result to Inf.
- Any number smaller than the smallest defined floating point number causes underflow. MATLAB assigns the result to 0.

End of Lecture 9

Please provide your feedback [▶ here](#)