

Numerical Methods

- Using R
 - Download R & R Studio
-

40% Final, 60% Assignments, Quizzes

Will consider: Solving "on Computer"

• $f(x) = 0$ $f: \mathbb{R} \rightarrow \mathbb{R}$ some fn

$$\underset{n \times n}{A} \underset{n \times 1}{x} = \underset{n \times 1}{b}$$

• $\int_a^b f(x) dx$ $f: \mathbb{R} \rightarrow \mathbb{R}$ bounded continuous

• $y: [0, T] \rightarrow \mathbb{R}$ $f: \mathbb{R}_+ \times \mathbb{R} \rightarrow \mathbb{R}$

$$\frac{dy}{dt} = f(t, y)$$

- often not able to find exact solns
- Need approximations for practical purposes
- How good is approximation ?
- What is good enough ?

Efficiency : How much work to reach soln.

Accuracy : How close to true value

Precision : Level of detail in our solution.

$$\pi = 3.1415926535\cdots$$

$$\frac{22}{7} = 3.14285714\cdots$$

3.1416
is more
accurate
approximation
to π

Computer softwares work with floating point numbers

Floating point numbers in R

64 bits

1 bit
sign

52 bits
mantissa

1 bit
sign 10 bits
exponent

$4 =$

$$1 \times 2^2$$

1
mantissa
↑
2
exponent

$$15 = 1111 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2 + 1 \times 2^0$$

sign 0 is +1 1 is -1



Binary

$$\underline{1.111} \times 2^3$$

Largest number that R can work
with = ?

$$\text{Max Exponent} = \underbrace{111\ldots1}_{10} = 2^{10} - 1 = 1023$$

Max

$$\text{Mantissa} = \underbrace{1.11\ldots1}_5 = \left[1 + 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + \dots + 1 \cdot \frac{1}{2^5} \right]$$

Binary

Decimal

$$\left(1 + \frac{1}{2} + \dots + \frac{1}{2^{51}}\right) \times 2^{1023} \leftarrow \begin{matrix} \text{Max \#} \\ R \text{ can handle} \end{matrix}$$

$$> 2^{1023} \quad NA$$

$$> 1.5 \times 2^{1023} \quad \checkmark$$

smallest positive number R can handle: $\frac{1}{2^{51}} \times 2^{-1023}$

$$> 20.55 - 19.2 - 1.35 \\ 1.332268 e^{-15}$$

$$> 20.55 - 1.35 - 19.2 \\ 0$$

Errors in computing

- Finite storage space
- Cannot store irrational numbers like π exactly (truncation error)

Effects

- Floating Point number line
 - largest & smallest number

- Need to be more careful.

Computer Storage

Bit (Binary digit) 0 or 1

Smallest unit of storage

Think of this as switching on/off.

Byte : Composed of 8 bits

Can store up to 2^8 characters
in a byte.

ASCII in Wikipedia : (Check)

Can store characters like 0, 1, 2, 3, ..., 9

A, B, C, ..., Z, a, b, c, ..., z,

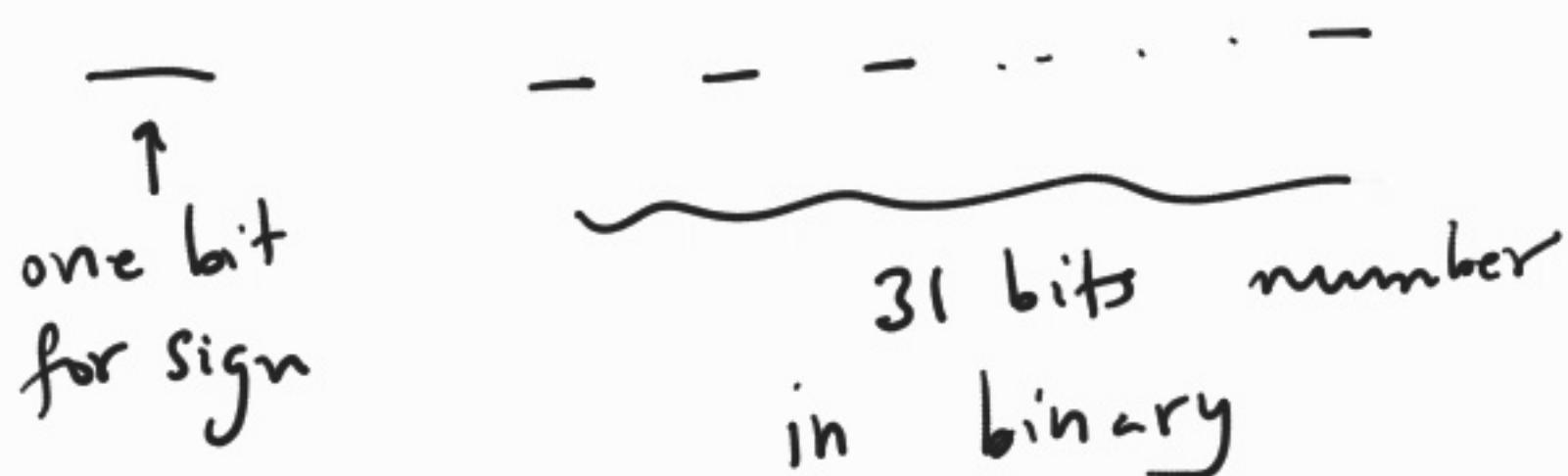
@, ?, ;, ... in a byte.

Computers can work with 32 bits

or 64 bits (Machine) in one go

Storing numbers

Integers stored in R using 32 bits



Decimal : 32 29 201 . . .

Binary : 101, 1111, 10, 110 . . .

$$32 = 3 \times 10^1 + 2 \times 10^0$$

$$201 = 2 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$$

Binary $101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$

most significant least significant digit in decimal

Binary $1110 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 $= 14$ in decimal.

Max number that can be stored
as an integer is

1 1 1 . . . 1
~~~~~  
31 bit

$$1 \times 2^{30} + \dots + 1 \times 2^1 + 1 \times 2^0$$
$$= 2^{31} - 1$$

In R eq. to store  $2^{31}$  as an integer  
 $> 2^{31} L$

## Floating point numbers - Default in R

Decimal  $21.8125$

$$= 2 \times 10^1 + 1 \times 10^0 + 8 \times 10^{-1} + 1 \times 10^{-2}$$

$$+ 2 \times 10^{-3} + 5 \times 10^{-4}$$

Binary of  $21.8125$

$$21 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0$$

$$= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1$$

$$+ 1 \times 2^0$$

21 in Binary  $10101$

$$0.8125 = 0.5 + 0.3125$$

$$= 0.5 + 0.25 + 0.0625$$

$$= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$$

$$+ 1 \times 2^{-4}$$

Binary  $0.1101$

$$2^{-1} = 0.5$$

$$2^{-2} = 0.25$$

$$2^{-3} = 0.125$$

$$2^{-4} = 0.0625$$

Binary of  $21.8125$  is

1 0 1 0 1 . 1 1 0 1

Eg      Decimal      0.2

$$\begin{aligned} 0.2 &= 0.125 + 0.075 \\ &= 0.125 + 0.0625 \\ &\quad + 0.0125 \\ &\leq 0.125 + 0.0625 \\ &\quad + 0.0078125 + \dots \end{aligned}$$

$$\begin{aligned} 2^{-1} &= 0.5 \\ 2^{-2} &= 0.25 \\ 2^{-3} &= 0.125 \\ 2^{-4} &= 0.0625 \\ 2^{-5} &= 0.03125 \\ 2^{-6} &= 0.015625 \\ 2^{-7} &= 0.0078125 \\ &\vdots \end{aligned}$$

? Infinite Expansion.

Remark. Infinite expansion <sup>in binary</sup> cannot be stored!

A floating point number is stored  
in 64 bits

Sign - 1 bit

Mantissa - 52 bits

Exponent - 11 bits (1 bit for sign)

64 bits

Sign Bit

0 or 1  
(positive)      (negative)

Exponent Range, [-1022, 1023]  
integers in

Stored in binary with a bias of 1023

So -1022 is stored as  $-1022 + 1023 = 1$   
1023                  "                   $1023 + 1023 = 2046$

$$-1022 \rightarrow 1$$

In 11 Bits

0 0 . . . 0 1  
~~~~~  
10 0's

$$\begin{aligned}1023 \rightarrow 2046 &= 1024 + 512 + 256 \\&+ 128 + 64 + 32 \\&+ 16 + 8 + 4 + 2\end{aligned}$$

= 1 1 1. . . 1 0
~~~~~  
10 1's

Remark. Note exponent -1022, . . . , 0  
with bias when stored in binary have 0 as 1st digit

1, 2, . . . , 1023 when stored with bias in binary have 1 as 1st digit

Mantissa      52 bits

$$231.56 \rightarrow 2.3156 \times 10^2$$

$$0.0156 \rightarrow 1.56 \times 10^{-2}$$

Similarly write binary number with  
just one | to the left of binary point.  
( get extra bit for free ! )

0.011011...1

54

= 10.1011...1 "x"  $2^2$

always 1

Eg  $21.8125 \rightarrow 10101.1101$  in binary

Sign Bit

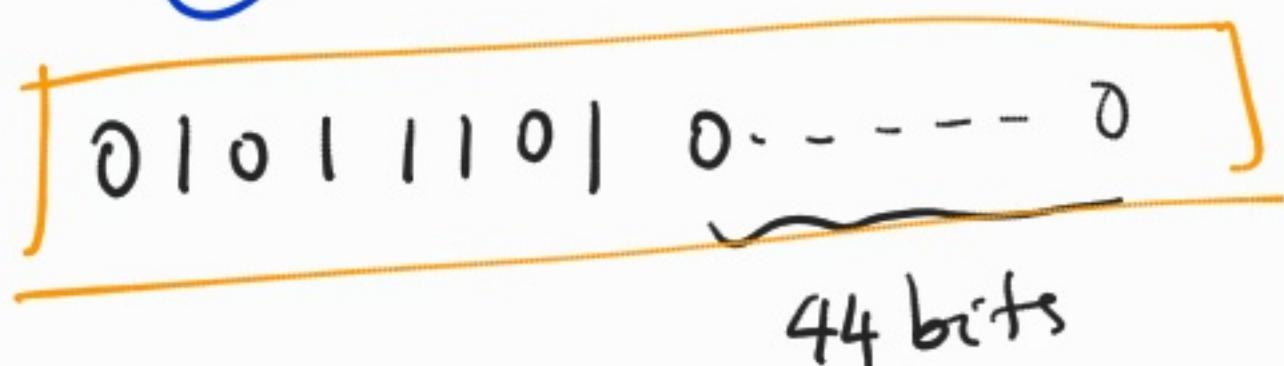


$$\textcircled{a} \leftarrow 10101.1101 = 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 \\ + 0 \times 2^1 + 1 \times 2^0 \\ + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$

$$\textcircled{b} \leftarrow 1.01011101 = 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ + \dots$$

$$\textcircled{a} = 2^4 \times \textcircled{b}$$

Mantissa :



Exponent :

$$4 \rightarrow 1027$$

$$= 1024 + 2 + 1$$



"Smallest +ve number"

Q. . . .  
1.00

$$\begin{array}{r} 0 \ 0 \ . \ - \ - \cdot 0 \ 1 \\ \hline & \uparrow \\ & \text{exponent} \end{array}$$

0 0 . . . 0  
—————  
52 bit  
memory

$$\left( 1 + \sum_{j=1}^{52} 0 \times 2^{-j} \right) \times 2^{1023}$$

>.Machine \$ double .xmin

Remark : R can work with  
smaller numbers ("denormal  
numbers")  
at the cost of  
precision. (some digits from  
mantissa go to  
exponent)

"Largest +ve number"

---

$$\begin{array}{r} 0 \quad \overbrace{111\ldots10}^{\text{1}} \\ \hline \overbrace{1111\ldots1}^{52 \text{ bit}} \end{array}$$

$$\left(1 + \sum_{i=1}^{52} 1 \times 2^{-i}\right) \times 2^{2046 - 1023}$$

Remark: R cannot work with  
larger numbers. losing precision  
not an option

---

## Special numbers

|              |                  |                  |    |
|--------------|------------------|------------------|----|
| 0            | 0 0 0 0 . . . 0  | 0 0 . . . 0      | +0 |
| 1            | 0 . . . . . 0    | 0 . . . 0        | -0 |
| wave<br>sign | wave<br>exponent | wave<br>mantissa |    |

|   |             |                 |         |
|---|-------------|-----------------|---------|
| 0 | 1. . . . .  | 0 . . . 0       | +Inf    |
| 1 | 1. . . . .  | 0 . . . . 0     | -Inf    |
| 0 | 1. . . . .  | 1 0 . . . 0     | Nan     |
| 1 | 1 . . . . . | { 0 . . . . 0 } |         |
|   |             |                 | 51 bits |

"NAN" not a number

Eg

$$1 = 1 \times 2^0 \xrightarrow{\text{Binary}} 1.00\ldots$$

Exponent

|      |   |      |   |          |        |
|------|---|------|---|----------|--------|
| Sign | 0 | 011. | 1 | Mantissa | 00...0 |
|------|---|------|---|----------|--------|

Exponent = 0  $\rightarrow 10^{23}$

---

---