Horner's scheme:

$$(a_n a_{n-1} \dots a_0)_{(b)} = a_0 + b(a_1 + b(a_2 + b(a_3 + \dots b a_n) \dots)$$

Thus, the following algorithm can be used for converting to base b:

1) Input: $x_{(10)}$
2) $i := 0$
3) while $x > 0$ do
4) $\quad a_i := x \bmod b$ rest of integer div
5) $\quad x := x \operatorname{div} b$ integer division
6) $\quad i := i + 1$
7) end while
8) output $a_n a_{n-1} \dots a_0$

Problems are introduced by the fact that computers treat/store numbers with finite precision.

<u>Def 9:</u> A <u>normalized floating point representation</u> wrt base b stores a number x as

$$x = 0.a_1 \dots a_k \cdot b^n$$

where $a_i \in \{0, 1, \dots, b-1\}$ are the <u>digits</u>, k is called <u>precision</u>, n is called <u>exponent</u>, $a_1, \dots, a_k$ is called <u>mantissa</u>, $a_1 \neq 0$. The fact that $a_1 \neq 0$ is called normalization, it makes the representation unique.

<u>Examples 10:</u> • base $b = 10$: 32.213

$\rightarrow 0.32213 \cdot 10^2$

- base $b=2$: $x = \pm 0.b_1 b_2 \dots b_K \cdot 2^n$,
  where $b_1 = 1$.

In such representation we have some peculiari

- adding numbers is commutative, i.e. $x+y=y+x$
- but not associative, i.e. $(x+y)+z \neq x+(y+z)$
  (not always true)

For example: Assume $k=6$:

$$\underbrace{1+1+1+\dots+1}_{1 \text{ million times}} + 1{,}000{,}000 = 2{,}000{,}000$$

the computer does:

$$1+1 = 0.1\cdot 10^1 + 0.1\cdot 10^1 = 0.2\cdot 10^1$$
$$1+1+1 = 0.2\cdot 10^1 + 0.1\cdot 10^1 = 0.3\cdot 10^1$$
$$\vdots$$
$$1 \text{ million} = 0.1\cdot 10^7$$

now we add $1{,}000{,}000 = 0.1\cdot 10^7$

so $0.1\cdot 10^7 + 0.1\cdot 10^7 = 0.2\cdot 10^7$ ✓

Now reverse the order of adding the numbers:

$$1{,}000{,}000 + \underbrace{1+1+\dots+1}_{1 \text{ million times}}$$

the computer does $1{,}000{,}000 + 1 = 0.1\cdot 10^7 + 0.1\cdot 10^1$

$$= 0.100\,000 \cdot 10^7 + 0.100\,000 \cdot 10^1$$
$$= 0.100\,000 \cdot 10^7 + 0.000\,000|1 \cdot 10^7$$

lost, becaus precision i only $k=6$

$$= 0.1 \cdot 10^7$$
$$= 1 \text{ million} \ ⚡$$

so, for the computer $1{,}000{,}000 + \underbrace{1+1+\dots+1}_{1 \text{ million times}} = 1{,}000{,}0$

we lose significant digits.

Avoid adding numbers with different orders of magnitude. Add numbers in increasing order of magnitude ... their size.

- Compute $x - \sin(x)$ for x close to 0, e.g. $x=$ ... assume $k = 10$ precision.

Then:
$$
\begin{aligned}
x &= 0.66666\ 66667 \cdot 10^{-1} \\
\sin(x) &= 0.66617\ 29492 \cdot 10^{-1} \\
\hline
x - \sin(x) &= 0.00049\ 37175 \cdot 10^{-1} \\
&= 0.49371\ 75\underbrace{000}_{} \cdot 10^{-4}
\end{aligned}
$$

no information is carried here, we are losing precision by 3 digits

> Avoid substracting numbers of similar size, because it leads to loss of precision!

A remedy is to consider the Taylor series expansion of $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} \pm$

So $x - \sin(x) = + \frac{x^3}{3!} - \frac{x^5}{5!} \mp \cdots$

With just 3 terms this yields: $0.49371\ 74328 \cdot 10^{-4}$ that has an error of $\leq 10^{-13}$

> <u>Theorem 11</u>: Let x, y be two normalized floating point numbers with $x > y > 0$ and base $b = 2$. If there exist $p, q \in \mathbb{N}_0$ such that
> $$2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$$
> then at most p and at least q significant digits (bits) are lost during substraction.

2 <u>linear systems of equations</u>

Def 12: A linear system of equations is given in the form $Ax = b$, $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}$
i.e. the matrix A has m rows and n cols, x is a vector with n unknowns, b has m entries thus the system has m equations.
Since the degree of all x_i is equal to one it is a <u>linear</u> equation. If $n = m$ the system is called square. We can also write

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad , i = 1, \dots, m$$

linear systems of equations arise, e.g.

- geometrical problems
- electrical circuits + Kirchhoff's laws / Ohm's law
- solving differential equations
- GPS

2.1 Gaussian Elimination (GE)

Assume that $m = n$, square system.
Idea of GE: Do row operations to produce an upper triangular matrix (echelon form). Then do back substitution to solve the system.

Row operations:

1) Swap rows
2) Scale rows, i.e. multiply by <u>scalar</u>
3) Add multiple of one row to another row

Example 13:

$$A = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix}, \quad b = \begin{bmatrix} 16 \\ 26 \\ -19 \\ -34 \end{bmatrix}$$

STEP 1:

Do GE in a systematic way:

- Augmented matrix

pivot element

pivot row (−2) $(-\frac{1}{2})$ (1)

$$\left[\begin{array}{cccc|c} \boxed{6} & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right]$$

what is the relevant factor: $-2 = -\frac{12}{6} = -\frac{\text{element to c}}{\text{pivot elem}}$

pivot element

pivot row (−3) $(\frac{1}{2})$

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & \boxed{-4} & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & 2 & 3 & 22 & -18 \end{array}\right]$$

...

$$\left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 0 & -3 & -3 \end{array}\right]$$ upper triangular form

STEP 2: Backward substitution:

- last equation: $-3x_4 = -3 \iff \boxed{x_4 = 1}$
- 2nd to last eq: $2x_3 - 5x_4 = -9$

  substitute x_4: $2x_3 - 5 = -9$

  $\iff \boxed{x_3 = -2}$

- ... finally $x_1 = 3$, $x_2 = 1$, $x_3 = -2$, $x_4 = 1$.

Algorithm 14:

1) Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$

Forward elimination:

2) For $k = 1, \dots, n-1$ for all pivot rows
3)   For $i = k+1, \dots, n$ for all rows below pivot row

For $j = k, \dots, n$ for all cols from pivot on

$a_{ij} := a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}$ $\quad a_{kk}$ = pivot el.

end for

$b_i := b_i - \frac{a_{ik}}{a_{kk}} b_k$ $\quad$ don't forget rhs

end for

end for

<u>Backward substitution</u>

$x_n = b_n / a_{nn}$

for $i = n-1, \dots, 1$

$sum := b_i$

for $j = n, \dots, i+1$

$sum := sum - a_{ij} x_j$

end

$x_i := sum / a_{ii}$

end for