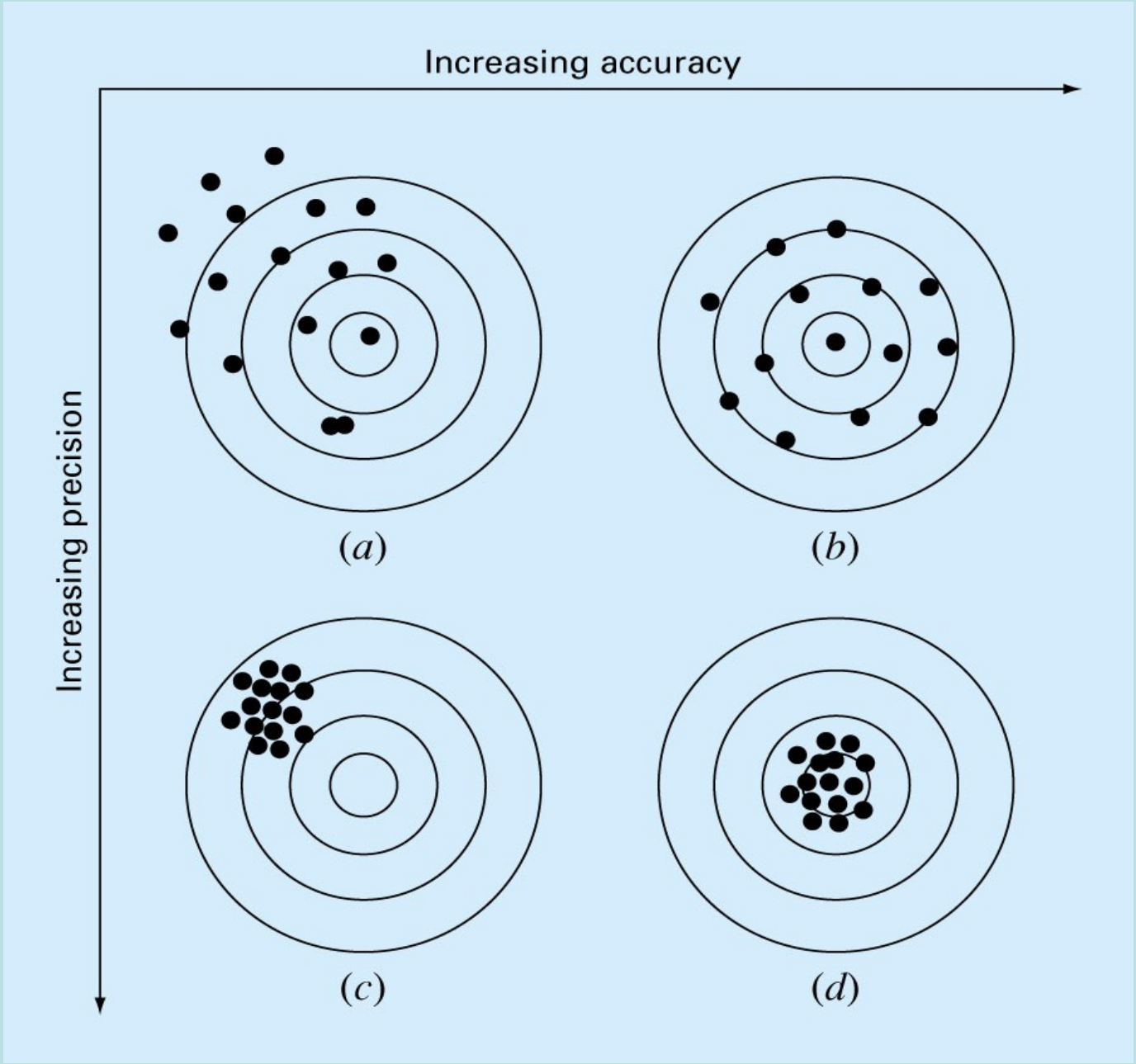
Errors

Approximations and Round-Off Errors

- For many engineering problems, we cannot obtain analytical solutions.
- Numerical methods yield approximate results, results that are close to the exact analytical solution. We cannot exactly compute the errors associated with numerical methods.
  - Only rarely given data are exact, since they originate from measurements. Therefore there is probably error in the input information.
  - Algorithm itself usually introduces errors as well, e.g., unavoidable round-offs, etc …
  - The output information will then contain error from both of these sources.
- How confident we are in our approximate result?
- The question is "*how much error is present in our calculation and is it tolerable?*"

- Accuracy. How close is a computed or measured value to the true value
- Precision (or *reproducibility*). How close is a computed or measured value to previously computed or measured values.
- Inaccuracy (or *bias*). A systematic deviation from the actual value.
- Imprecision (or *uncertainty*). Magnitude of scatter.

Fig. 3.2



Significant Figures

- Number of significant figures indicates precision. Significant digits of a number are those that can be *used* with *confidence,* e.g., the number of certain digits plus one estimated digit.

53,800 How many significant figures?

5.38×10^4	3
5.380×10^4	4
5.3800×10^4	5

Zeros are sometimes used to locate the decimal point not significant figures.

0.00001753	4
0.0001753	4
0.001753	4

Error Definitions

True Value = Approximation + Error

= True value – Approximation (+/-)

True error

$$\text{True fractional relative error} = \frac{\text{true error}}{\text{true value}}$$

$$\text{True percent relative error, } \varepsilon_t = \frac{\text{true error}}{\text{true value}} \times 100\%$$

- For numerical methods, the true value will be known only when we deal with functions that can be solved analytically (simple systems). In real world applications, we usually not know the answer a priori. Then

$$\varepsilon_a = \frac{\text{Approximate error}}{\text{Approximation}} \times 100\%$$

- *Iterative approach,* example Newton's method

$$\varepsilon_a = \frac{\text{Current approximation - Previous approximation}}{\text{Current approximation}} \times 100\%$$

(+ / -)

- Use absolute value.
- Computations are repeated until stopping criterion is satisfied.

$$|\varepsilon_a| < \varepsilon_s$$

Pre-specified % tolerance based on the knowledge of your solution

- If the following criterion is met

$$\varepsilon_s = (0.5 \times 10^{(2-n)})\%$$

you can be sure that the result is correct to at least n significant figures.

Round-off Errors

- Numbers such as π, e, or $\sqrt{7}$ cannot be expressed by a fixed number of significant figures.
- Computers use a base-2 representation, they cannot precisely represent certain exact base-10 numbers.
- Fractional quantities are typically represented in computer using "floating point" form, e.g.,

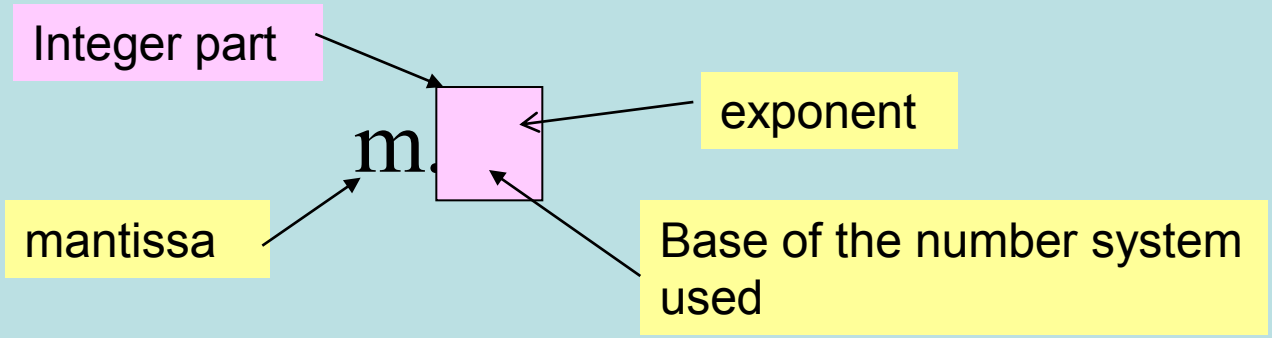


Figure 3.3

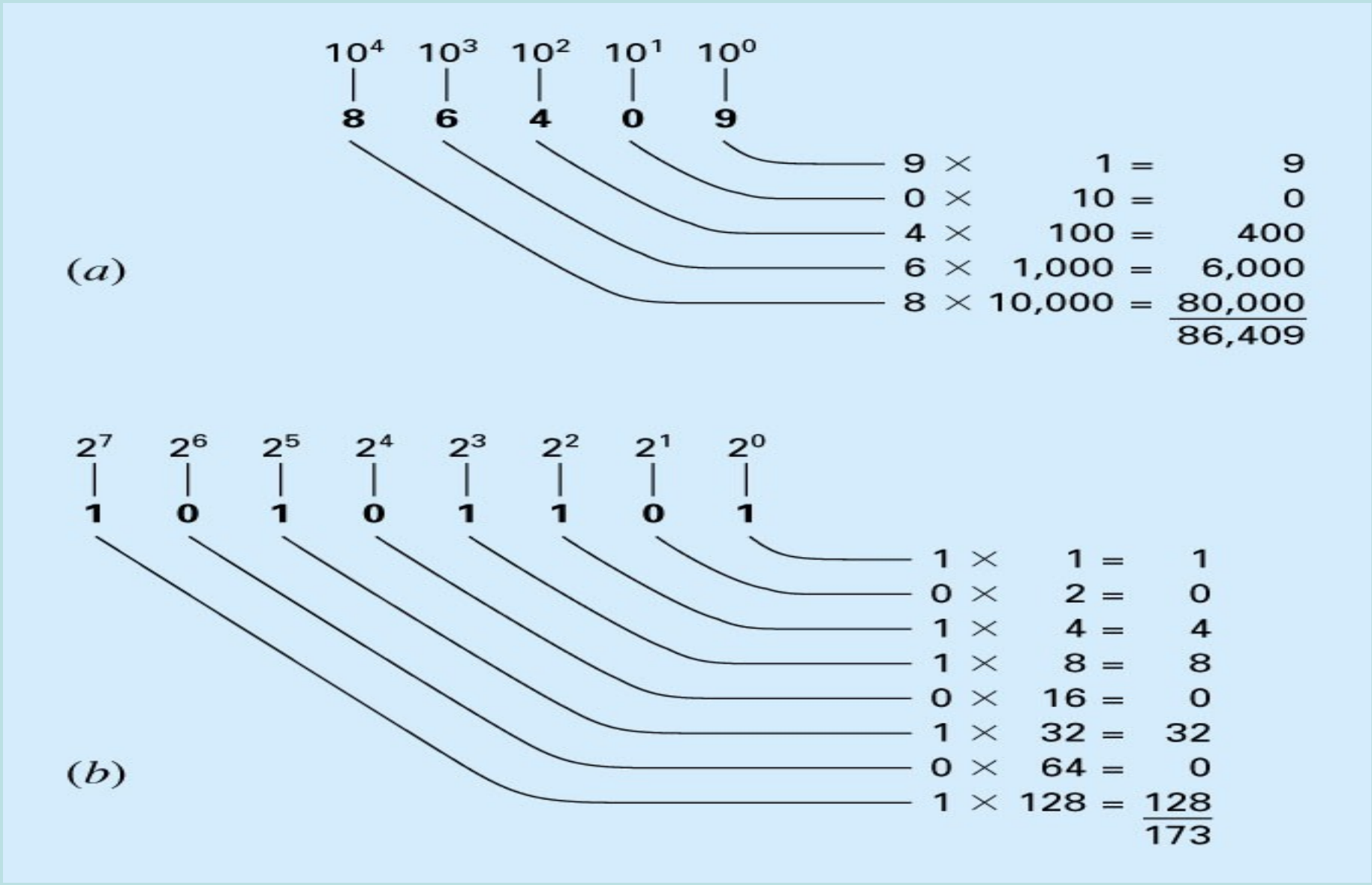

10^4 10^3 10^2 10^1 10^0
8 6 4 0 9
9 × 1 = 9
0 × 10 = 0
4 × 100 = 400
6 × 1,000 = 6,000
8 × 10,000 = 80,000
86,409
(a)
2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
1 0 1 0 1 1 0 1
1 × 1 = 1
0 × 2 = 0
1 × 4 = 4
1 × 8 = 8
0 × 16 = 0
1 × 32 = 32
0 × 64 = 0
1 × 128 = 128
173
(b)

Figure 3.4

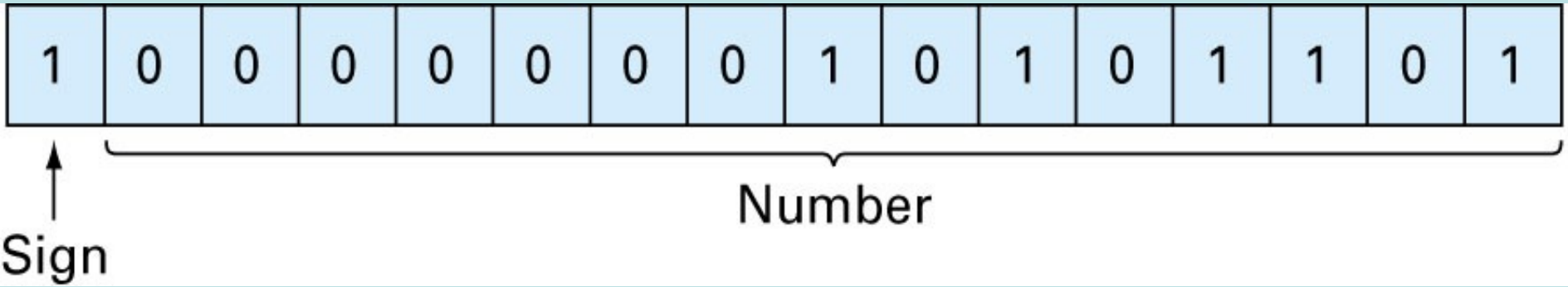
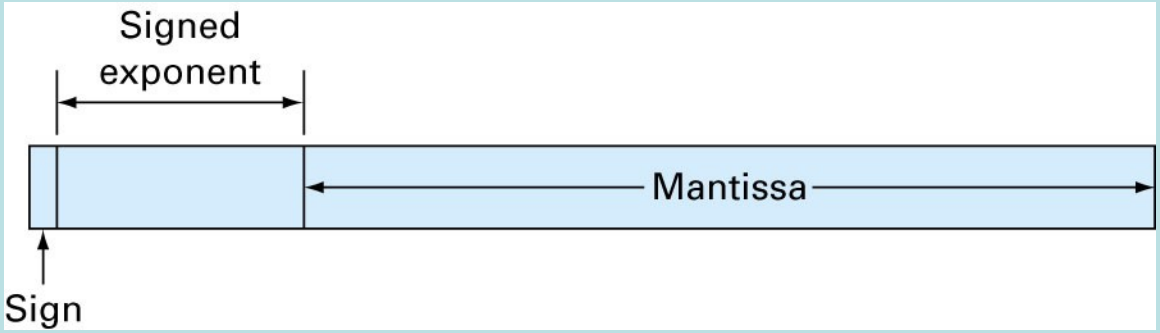


Figure 3.5



156.78 ⏩ 0.15678×10^3 in a floating point base-10 system

$$\frac{1}{34} = 0.029411765$$

Suppose only 4 decimal places to be stored

$$0.0294 \times 10^0$$

$$\frac{1}{2} \le |m| < 1$$

- Normalized to remove the leading zeroes. Multiply the mantissa by 10 and lower the exponent by 1

0.2941×10^{-1}

Additional significant figure is retained

$$\frac{1}{b} \leq |m < 1|$$

Therefore

for a base-10 system $0.1 \leq m < 1$

for a base-2 system $0.5 \leq m < 1$

- Floating point representation allows both fractions and very large numbers to be expressed on the computer. However,
  - Floating point numbers take up more room.
  - Take longer to process than integer numbers.
  - Round-off errors are introduced because mantissa holds only a finite number of significant figures.

Chopping

Example:

π=3.14159265358 to be stored on a base-10 system carrying 7 significant digits.

π=3.141592 chopping error ε_t=0.00000065

If rounded

π=3.141593 ε_t=0.00000035

- Some machines use chopping, because rounding adds to the computational overhead. Since number of significant figures is large enough, resulting chopping error is negligible.