Emory University
MATH 315 Numerical Analysis
Learning Notes

Jiuru Lyu

September 11, 2023

Contents

1 Floating Point Numbers

1.1 Binary Representation

Definition 1.1.1 (Binary). 0 and 1; on and off.

Example 1.1.2 Represent Numbers in Base-2

Consider $13 = 1(10) + 3(1) = 1(10) + 3(10^0)$ in base-10. It can be converted into base-2 by decomposing 13 as $1(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$.

Example 1.1.3 Fractions in Base-2

$$\frac{7}{16} = \frac{1}{16}(7) = \left(2^{-4}\right)\left(2^2 + 2^1 + 2^0\right) = 2^{-2} + 2^{-3} + 2^{-4}.$$

Example 1.1.4 Repeating Fractions in Base-2

$$\begin{aligned} \frac{1}{5} = \frac{1}{8} + \varepsilon_1 \quad &\implies \quad \varepsilon_1 = \frac{1}{5} - \frac{1}{8} = \frac{8-5}{(5\times 8)} = \frac{3}{40} \\ \varepsilon_1 = \frac{3}{3(16)} + \varepsilon_2 \quad &\implies \quad \cdots \end{aligned}$$

Repeating the steps above, we would finally get

$$\frac{1}{5} = \frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} + \cdots$$

Theorem 1.1.5

Let $n \in \mathbb{Z}$ and $n \geq 1$, then

$$\sum_{k=0}^{n-1} 2^k = 2^{n-1} + 2^{n-2} + \cdots + 2^0 = 2^n - 1.$$

1.2 Integers in Computers

Definition 1.2.1 (Storing Integers). `unit8` stands for unsigned integers and `int8` stands for signed integers.

Remark. *The 8 here represents 8 bits. It is a measure of how much storage (how many 0s or 1s).*

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
unsigned:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
signed:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Example 1.2.2

$$\texttt{unit8}(13) = 00001101$$

Since $-13 = 1(-2^7) + 1(2^6) + 1(2^5) + 1(2^4) + 0(2^3) + 0(2^2) + 1(2^1) + 1(2^0)$, we have

$$\texttt{int8}(-13) = 11110011$$

Remark. *Largest and Smallest Integers:*

$$\begin{aligned}
\texttt{uint8}(x_L) &= 11111111 &&\Longrightarrow x_L = 2^7 + 2^6 + \cdots + 2^0 = 2^8 - 1 = 255\\
\texttt{uint8}(x_S) &= 00000000 &&\Longrightarrow x_S = 0(2^7) + 0(2^6) + \cdots + 0(2^0) = 0\\
\texttt{int8}(x_L) &= 01111111 &&\Longrightarrow x_L = 0(-2^7) + 2^6 + \cdots + 2^0 = 2^7 - 1 = 127\\
\texttt{int8}(x_S) &= 100000000 &&\Longrightarrow x_S = 1(-2^7) + 0(2^6) + \cdots + 0(2^0) = -128
\end{aligned}$$

1.3 Representation of Floating Point Numbers

Definition 1.3.1 (Normalized Scientific Notation). Only 1 digit (non-zero) to the left of the decimal point.

Example 1.3.2

$$\begin{aligned}
&123.456 \times 10^7\\
&12.3456 \times 10^8\\
&1.23456 \times 10^9 \rightarrow \text{normalized}
\end{aligned}$$

Definition 1.3.3 (Anatomy of Floating Point Numbers). A floating point number, $\texttt{float}(x)$, consists of three parts: $s(x)$ (sign bit), $e(x)$ (exponent bits), and $f(x)$ (fraction bits).

Definition 1.3.4 (Precision). Precision is defined by the number of bits per part:

	$s(x)$	$e(x)$	$f(x)$	total
double precision (DP)	1	11	52	64
single precision (SP)	1	8	23	32
half precision (HP)	1	5	10	16

Remark. *The less bits the float point number has, the less storage it requires and faster computation it performs, but more error introduces.*

Definition 1.3.5 (Floating Point Number).

$$\texttt{float}(x) = (-1)^{s(x)}\left(1 + \frac{f(x)}{2^{\text{\# of fraction bits}}}\right)2^{E(x)}, \tag{1}$$

where $E(x)$ is called the *unbiased exponent* because it is centered about 0 and is calculated through the $e(x)$, the *biased exponent* because it can only be non-negative integers, by the following formula:

$$E(x) = e(x) - \left(2^{\text{\# of exponent bits}-1} - 1\right).$$

Remark. *Eq. (1) is in normalized scientific notation because the largest number $f(x)$ can represent is $2^{\text{\# of fraction bits}} - 1$. Hence,*

$$1 + \frac{f(x)}{2^{\text{\# of fraction bits}}} < 2,$$

and thus there will be only 1 digit in front of the decimal point.

Example 1.3.6 Formula for a Floating Point Number in Double Precision (DP)

$$\texttt{float}_{\text{DP}}(x) = (-1)^{s(x)}\left(1 + \frac{f(x)}{2^{52}}\right)2^{e(x)-1023}.$$

Example 1.3.7 Converting DP into Decimal

Suppose a DP floating number is stored as $s(x) = 0$, $e(x) = 10000000011$, and $f(x) = 0100100\cdots0$. Find its representation in decimal base-10.

Solution 1.

$e(x) = 10000000011 = 2^{10} + 2^1 + 2^0$ and $f(x) = 0100100\cdots0 = 2^{50} + 2^{47}$. Then, the unbiased exponent $E(x) = e(x) - 1023 = 2^{10} + 2^1 + 2^0 - (2^{10} - 1) = 4$. So,

$$\begin{aligned}
\texttt{float}_{\text{DP}}(x) &= (-1)^{s(x)} + \left(1 + \frac{f(x)}{2^{52}}\right)2^{E(x)} \\
&= (-1)^0\left(1 + \frac{2^{50} + 2^{47}}{2^{52}}\right)2^4 \\
&= \left(1 + 2^{-2} + 2^{-5}\right)2^4 \\
&= 2^4 + 2^2 + 2^{-1} \\
&= 16 + 4 + 0.5 = 20.5
\end{aligned}$$

□

Example 1.3.8 Converting Value to DP

Suppose a number in base-10 is -10.75. Find its representation of floating point number under DP.

Solution 2.

We have

$$\begin{aligned}\texttt{value}(x) = -10.75 &= (-1)(10+0.75)\\ &= (-1)\big(2^3+2^1+2^{-1}+2^{-2}\big)\\ &= (-1)\big(1+2^{-2}+2^{-4}+2^{-5}\big)2^3 \quad \Big[\text{In normalized scientific notation}\Big]\\ &= (-1)^1\left(1+\frac{2^{50}+2^{48}+2^{47}}{2^{52}}\right)2^{1026-1023}\\ &= (-1)^1\left(1+\frac{2^{50}+2^{48}+2^{47}}{2^{52}}\right)2^{2^{10}+2^1-1023}\end{aligned}$$

So, we have $s(x) = 1$, $e(X) = 10000000010$, and $f(x) = 010110\cdots0$. □

Theorem 1.3.9 Some Special Rules

1. The formula
$$\texttt{value}(x) = (-1)^{s(x)} + \left(1+\frac{f(x)}{2^{52}}\right)2^{e(x)-1023}$$
only holds when $0 < e(x) < 2^{11}-1$ or $00\cdots01 < e(x) < 11\cdots10$.

2. If $e(x) = 11\cdots1$, then it encodes special numbers.

3. If $e(x) = 00\cdots0$:
   - If $f(x) = 00\cdots0$, then $\texttt{value}(x) = 0$.
   - If $f(x) > 0$, it encodes a *denormalized floating point number*:
   $$\texttt{value}(x) = (-1)^{s(x)}\left(0+\frac{f(x)}{2^{52}}\right)2^{-1022}.$$
   This denormalized floating point number is more precise when describing really small things.

Definition 1.3.10 (Machine Epsilon/ε_{WP}). Let “WP” stands for the working precision (DP/SP/HP/etc.). The *machine epsilon,* denoted as ε_{WP}, is the gap between 1 and the next largest floating point number. Equivalently, it can be viewed as the smallest possible non-zero value of $\frac{f(x)}{2^{\text{number of fraction bits}}}$. So, $\varepsilon_{\text{DP}} = 2^{-52}$, $\varepsilon_{\text{SP}} = 2^{-23}$, and $\varepsilon_{\text{HP}} = 2^{-10}$.

Definition 1.3.11 (Special Numbers).

1. ±0: when $s(x) = \pm1$ and $e(x) = f(x) = 0$.

2. ±`Inf`

3. `NaN`: not-a-number

Definition 1.3.12 (Floating Point Arithmetic).

1. The set of real numbers, $\mathbb{R}$, is closed under arithmetic operations.

2. The set of all WP floating point numbers, however, is not closed under arithmetic operations. For example, $\texttt{float}_{\text{DP}}(x) = \texttt{float}_{\text{DP}}(y) = 2^{52}+1$, but $xy = 2^{104} + \varepsilon$ cannot be represented using DP.

3. Suppose x and y are floating point numbers, then $x \oplus y = \texttt{float}(x+y)$ and $x \otimes y = \texttt{float}(xy)$. Consider `float` as a rounding process, we can also define subtraction and division of floating point numbers.

Example 1.3.13

Assume we are only allowed three significant digits (in Base-10) in a computer. Suppose $x = 1.23 \times 10^4$ and $y = 6.54 \times 10^3$. Find $x \oplus y = \texttt{float}(x+y)$.

Solution 3.

$$\begin{aligned} x \oplus y &= \texttt{float}(x+y) \\ &= \texttt{float}(1.23 \times 10^4 + 6.54 \times 10^3) \\ &= \texttt{float}(1.23 \times 10^4 + 0.654 \times 10^3) \\ &= \texttt{float}(1.884 \times 10^4) \\ &= 1.88 \times 10^4. \end{aligned}$$

□

1.4 Errors

Definition 1.4.1 (Errors We May See).

1. *Overflow*: The exponent is too large. This means $|x|$ is large and the computer will represent it as ±`Inf`. Note: In DP, $x_{\text{large}} = (2 - 2^{-52}) \times 2^{1023} \approx 1.798 \times 10^{308}$. This number is referred as `realmax` in MATLAB.

2. *Underflow*: Large negative exponent. This means $|x|$ is tiny and the computer will represent it as ± 0. Note: In SP, $x_{\text{small}} \approx 2.225 \times 10^{-53}$ and is referred as `realmin` in MATLAB.

3. *Roundoff error*: cutoff or round at some point.

Note that sometimes we are encounter the catastrophic cancellation, meaning the subtraction leads to our loss of significance or information. In this case, it is different from underflow error or roundoff error.

Example 1.4.2 Catastrophic Cancellation/Loss of Significance Due to Subtraction

$$\begin{aligned} x &= 3.141592920353983 \approx \frac{355}{113} && \text{16 digits} \\ y &= 3.141592653589794 \approx \pi && \text{16 digits} \\ x - y &= 0.000000266764189 && \text{9 digits} \end{aligned}$$

Definition 1.4.3 (Relative Error). Let $z \in \mathbb{R}$. The relative error between $\texttt{float}(z)$ and z is denoted as μ and

$$\begin{aligned} \mu &= \frac{\texttt{float}(z) - z}{z} \\ \texttt{float}(z) &= z(1+\mu), \end{aligned}$$

where we know

$$|\mu| \leq \frac{\varepsilon_{\text{WP}}}{2}.$$

Example 1.4.4 Propagation of Errors

There are two major sources of errors: storing number and arithmetics.

Consider a computer only allow 3 significant figures. Then $\varepsilon_{\text{WP}} = 0.01$.

Consider $x = \frac{1}{3}$, $y = \frac{8}{7}$, and $x + y = \frac{31}{21}$. Then,

$$\texttt{float}(x) = 0.333 = 3.33 \times 10^{-1} = x(1 + \mu_x).$$

Solving for μ_x:

$$\begin{aligned} \frac{333}{1000} &= \frac{1}{3}(1 + \mu_x) \\ \mu_x &= \frac{999}{1000} - 1 = \frac{-1}{1000} = -0.001 \end{aligned}$$

Note that $|\mu_x| = 0.01 < \frac{\varepsilon_{\text{WP}}}{2}$. Similarly, we can solve $\texttt{float}(y) = 1.14 \times 10^0 = y(1 + \mu_y)$ for $|\mu_y| = 0.0025$. Now, consider the floating point addition

$$\begin{aligned} x \oplus y &= \texttt{float}(\texttt{float}(x) + \texttt{float}(y)) \\ &= \texttt{float}(3.33 \times 10^{-1} + 1.14 \times 10^0) \\ &= \texttt{float}(1.473 \times 10^0) \\ &= 1.47 \times 10^0. \end{aligned}$$

Also, solve $x \oplus y = (x+y)(1+\mu_a)$ for $|\mu_a| = 0.0042$. Note that

$$|\mu_x| + |\mu_y| = 0.0035 < 0.0042 = |\mu_a|.$$

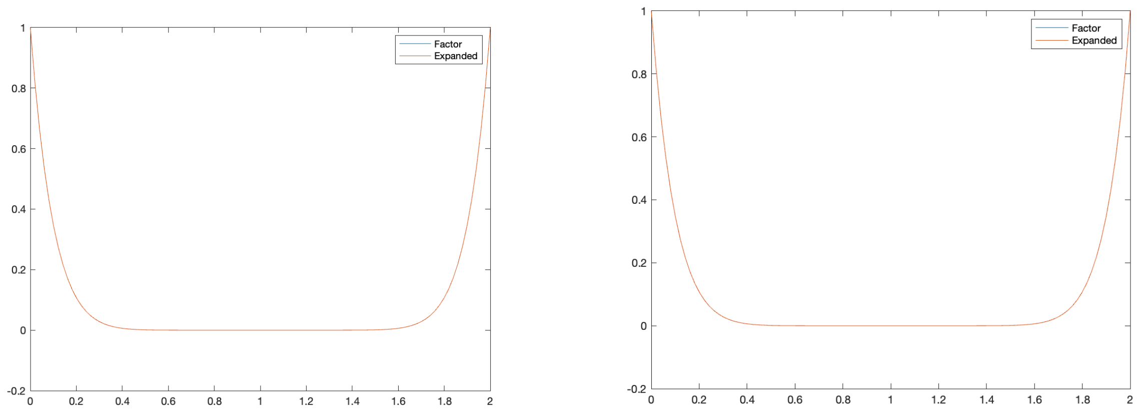
This is called the propagation of error.

Example 1.4.5 Plotting Exponentials Using Factored and Expanded Forms

Consider $p(x) = (1-x)^{10}$ and its expanded form. Plot them to see which is better.

Example 1.4.5

```matlab
%% Defining the Functions
p_1 = @(x) (1-x).^10;
p_2 = @(x) x.^10-10*x.^9+45*x.^8-120*x.^7+210*x.^6-252*x.^5+...
       210*x.^4-120*x.^3+45*x.^2-10*x+1;
%% Ploting the Functions
x = linspace(0, 2, 100);
plot(x, p_1(x))
hold on
plot(x, p_2(x))
legend("Factor", "Expanded")
%% Zooming In
y = linspace(0.99, 1.01, 100);
hold off
plot(y, p_1(y))
hold on
plot(y, p_2(y))
legend("Factor", "Expanded")
```



(a) Plotting Functions

(b) Zooming In

It seems that the two functions are the same (Fig 1(a)); however if we zooming in (Fig 1(b)), the expanded version introduces more error than the factored version because the expanded version requires more arithmetical operations in it.

2 Solutions of Linear Systems

Remark. *Assumption throughout this chapter:* $\mathbf{A}$ *is a square* $n \times n$ *matrix.*

2.1 Simply Solved Linear Systems

Definition 2.1.1 (Linear System).

- Equation form: x_i are variables (what we solve for) and a_{ij} are coefficients:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

  This system has n equations and n variables.

- Matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \implies \mathbf{A}x = b,$$

  where A is the coefficient matrix, a $n \times n$ matrix, x is the unknown, the solution vector with length n, and b is the right hand side, vector with length n.

Theorem 2.1.2 Number of Solutions to a Linear System
A linear system $\mathbf{A}x = b$ could have the following numbers of solutions:

- One unique solution: $\mathbf{A}x = b$ is nonsingular; $\mathbf{A}$ is invertible.
- No solutions: $\mathbf{A}x = b$ is singular.
- Infinite many solutions: $\mathbf{A}x = b$ is singular.

Theorem 2.1.3 Matrix-Vector Multiplication
Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$.

- View 1: Row-wise. Let $y = \mathbf{A}x$, then $y_i = \sum_{j=1}^{n} a_{ij}x_j$ as the i^{th} row of y.
- View 2: Column-wise. $\mathbf{A}x$ is a linear combination of columns of $\mathbf{A}$. So, $y = \sum_{j=1}^{n} x_j \vec{\mathbf{a_j}}$, where we regard A as $\begin{bmatrix} \vec{\mathbf{a_1}} & \vec{\mathbf{a_2}} & \cdots & \vec{\mathbf{a_n}} \end{bmatrix}$

Row-Wise Vector Multiplication

```
y = zeros(n, 1);
for i = 1:n % loop over rows
    for j = 1:n % loop over sum
        y(i) = y(i) + A(i,j) + x(j);
    end
end
```

Row-Wise Vector Multiplication (Vectorization)

```
y = zeros(n, 1);
for i = 1:n % loop over rows
    y(i) = A(i,:) * x(i); % vectorization
end
```

Column-Wise Vector Multiplication

```
y = zeros(n, 1);
for j = 1:n % loop over columns
    y = y + x(j) * A(:, j);
end
```

Definition 2.1.4 (Important Part of a Matrix).

- Diagonal Part
- Strictly Upper Triangular Part
- Strictly Lower Triangular Part

Theorem 2.1.5 Solving Diagonal Matrix

Given

$$\begin{bmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix},$$

we have

$$a_{11}x_1 = b_1; \quad a_{22}x_2 = b_2; \quad \cdots \quad a_{nn}x_n = b_n$$

So we have

$$x_i = \frac{b_i}{a_{ii}},$$

only if $a_{ii} \neq 0$.

Remark. $a_{ii} \neq 0$ *holes if* $\mathbf{A}$ *is invertible.*

Remark. *A Diagonal matrix is also a lower triangular matrix or an upper triangular matrix.*

Solving Diagonal Matrix

```
x = zeros(n, 1);
for i = 1:n
    x(i) = b(i) / A(i,i); % overflow and underflow
end
```

Theorem 2.1.6 Solving Lower Triangular Systems

Given

$$\begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ \vdots & & \ddots \\ a_{n1} & \cdots & & a_{nn} \end{bmatrix} \begin{bmatrix} x_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ v_n \end{bmatrix},$$

we have

$$\begin{aligned} a_{11}x_1 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

We can use the Forward Substitution to solve:

$$x_i = \frac{b_1 - a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{i(i-1)}x_{i-1}}{a_{ii}.}$$

Algorithm 1: Row-Oriented Forward Substitution

Input: matrix A $= [a_{ij}]$; vector $b = [b_i]$
Output: solution vector $x = [x_i]$

```
begin
    for i = 1 to n do // loop over rows

        for j = 1 to i-1 do // loop over columns

            $b_i := b_i - a_{ij}x_j$;
        $x_i := b_i / a_{ii}$;
```

Example 2.1.7

Given

$$\begin{bmatrix} -5 & & \\ 3 & 3 & \\ 2 & -5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix}.$$

Use column-wise forward substitution to solve this system.

Solution 1.

In column-wise:

$$x_1 \begin{bmatrix} -5 \\ 3 \\ 2 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix}.$$

1. Step 1: Solve for $x_1 = -10/-5 = 2$.

2. Step 2: Plug $x_1 = 2$ into the equation:

$$x_2 \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix} - (2) \begin{bmatrix} -5 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 17 \end{bmatrix}.$$

3. Step 3: Solve for $x_2 = -3/3 = -1$.

4. Step 4: Plug $x_2 = -1$ into the equation:

$$x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 17 \end{bmatrix} - (-1) \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 12 \end{bmatrix}.$$

5. Step 5: Solve for $x_3 = 12/4 = 3$.

□

Algorithm 2: Column-Oriented Forward Substitution

```
Input: matrix A = $[a_{ij}]$; vector $b = [b_i]$
Output: solution vector $x = [x_i]$
begin
    for j = 1 to n do
        $x_j := b_j/a_{jj}$;
        for i = j+1 to n do
            $b_i := b_i - a_{ij}x_j$;
```

Theorem 2.1.8 Computational Cost of Forward Substitution

Number of floating point operations $(+, -, \times, /)$ in row i is 1 division, $(i-1)$ multiplications, and $(i-1)$ subtractions. So, Number of floating points operations, or flops, of the algorithm is

$$\begin{aligned} \text{flops} = \sum_{i=1}^{n}(1+i-1+i-1) &= \sum_{i=1}^{n}(2i-1) \\ &= 2\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 \\ &= 2\left[\frac{(n+1)(n)}{2}\right] - n \\ &= n^2 \end{aligned}$$

Remark. *It should be the same number of flops if we do column-oriented forward substitution.*