

Emory University

MATH 315 Numerical Analysis

Learning Notes

Jiuru Lyu

September 18, 2023

Contents

1	Floating Point Numbers	2
1.1	Binary Representation	2
1.2	Integers in Computers	2
1.3	Representation of Floating Point Numbers	3
1.4	Errors	6
2	Solutions of Linear Systems	10
2.1	Simply Solved Linear Systems	10
2.2	GEPP and Matrix Factorization	14
2.3	Measuring Accuracy of Solutions	17

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

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

Example 1.1.4 Repeating Fractions in Base-2

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

Repeating the steps above, we would finally get

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

Theorem 1.1.5

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

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

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

$$\text{uint8}(13) = 00001101$$

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

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

Remark. *Largest and Smallest Integers:*

$$\text{uint8}(x_L) = 11111111 \implies x_L = 2^7 + 2^6 + \dots + 2^0 = 2^8 - 1 = 255$$

$$\text{uint8}(x_S) = 00000000 \implies x_S = 0(2^7) + 0(2^6) + \dots + 0(2^0) = 0$$

$$\text{int8}(x_L) = 01111111 \implies x_L = 0(-2^7) + 2^6 + \dots + 2^0 = 2^7 - 1 = 127$$

$$\text{int8}(x_S) = 100000000 \implies x_S = 1(-2^7) + 0(2^6) + \dots + 0(2^0) = -128$$

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

$$123.456 \times 10^7$$

$$12.3456 \times 10^8$$

$$1.23456 \times 10^9 \rightarrow \text{normalized}$$

Definition 1.3.3 (Anatomy of Floating Point Numbers). A floating point number, $\text{float}(x)$, 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).

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

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:

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

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)

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

Example 1.3.7 Converting DP into Decimal

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

Solution 1.

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

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

□

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

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

So, we have $s(x) = 1$, $e(x) = 10000000010$, and $f(x) = 010110 \dots 0$. □

Theorem 1.3.9 Some Special Rules

1. The formula

$$\text{value}(x) = (-1)^{s(x)} + \left(1 + \frac{f(x)}{2^{52}} \right) 2^{e(x)-1023}$$

only holds when $0 < e(x) < 2^{11} - 1$ or $00 \dots 01 < e(x) < 11 \dots 10$.

2. If $e(x) = 11 \dots 1$, then it encodes special numbers.

3. If $e(x) = 00 \dots 0$:

- If $f(x) = 00 \dots 0$, then $\text{value}(x) = 0$.
- If $f(x) > 0$, it encodes a *denormalized floating point number*:

$$\text{value}(x) = (-1)^{s(x)} \left(0 + \frac{f(x)}{2^{52}} \right) 2^{-1022}.$$

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/H-P/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 $s(x) = \pm 1$ and $e(x) = f(x) = 0$.

2. $\pm\text{Inf}$
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, $\text{float}_{\text{DP}}(x) = \text{float}_{\text{DP}}(y) = 2^{52} + 1$, but $xy = 2^{104} + \varepsilon$ cannot be represented using DP.
3. Suppose x and y are floating point numbers, then $x \oplus y = \text{float}(x + y)$ and $x \otimes y = \text{float}(xy)$. 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 $x \oplus y = \text{float}(x + y)$.

Solution 3.

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

□

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 $\pm\text{Inf}$. 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 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

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

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

$$\mu = \frac{\text{float}(z) - z}{z}$$

$$\text{float}(z) = z(1 + \mu),$$

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,

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

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 $\text{float}(y) = 1.14 \times 10^0 = y(1 + \mu_y)$ for $|\mu_y| = 0.0025$. Now, consider the floating point addition

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

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

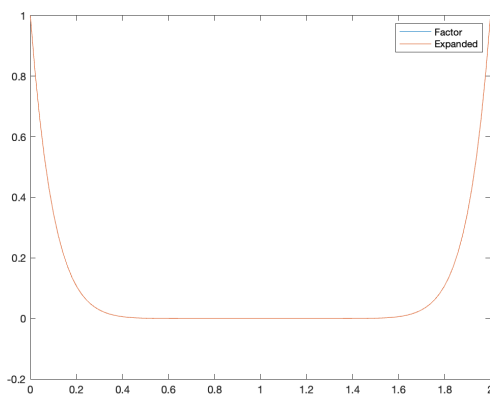
Consider $p(x) = (1 - x)^{10}$ and its expanded form. Plot them to see which is better.

Example 1.4.5

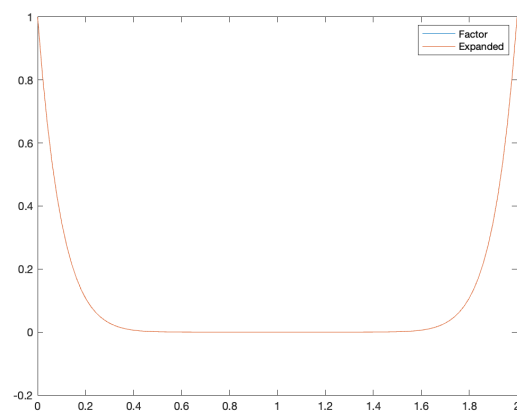
```

1  %% Defining the Functions
2  p_1 = @(x) (1-x).^10;
3  p_2 = @(x) x.^10-10*x.^9+45*x.^8-120*x.^7+210*x.^6-252*x.^5+...
4      210*x.^4-120*x.^3+45*x.^2-10*x+1;
5  %% Plotting the Functions
6  x = linspace(0, 2, 100);
7  plot(x, p_1(x))
8  hold on
9  plot(x, p_2(x))
10 legend("Factor", "Expanded")
11 %% Zooming In
12 y = linspace(0.99, 1.01, 100);
13 hold off
14 plot(y, p_1(y))
15 hold on
16 plot(y, p_2(y))
17 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: 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:

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

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 $Ax = b$ could have the following numbers of solutions:

- One unique solution: $Ax = b$ is nonsingular; A is invertible.
- No solutions: $Ax = b$ is singular.
- Infinite many solutions: $Ax = b$ is singular.

Theorem 2.1.3 Matrix-Vector Multiplication

Let $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$.

- View 1: Row-wise. Let $y = Ax$, then $y_i = \sum_{j=1}^n a_{ij}x_j$ as the i^{th} row of y .
- View 2: Column-wise. Ax is a linear combination of columns of A . So, $y = \sum_{j=1}^n x_j \vec{a}_j$, where we regard A as $\begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \cdots & \vec{a}_n \end{bmatrix}$

Row-Wise Vector Multiplication

```

1 y = zeros(n, 1);
2 for i = 1:n % loop over rows
3     for j = 1:n % loop over sum
4         y(i) = y(i) + A(i,j) * x(j);
5     end
6 end

```

Row-Wise Vector Multiplication (Vectorization)

```

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

```

Column-Wise Vector Multiplication

```

1 y = zeros(n, 1);
2 for j = 1:n % loop over columns
3     y = y + x(j) * A(:, j);
4 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. $a_{ii} \neq 0$ holds if \mathbf{A} is invertible.

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

Solving Diagonal Matrix

```

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

```

Theorem 2.1.6 Solving Lower Triangular Systems

Given

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

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:

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

Algorithm 1: Row-Oriented Forward Substitution

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

Output: solution vector $x = [x_i]$

```

1 begin
2   for i = 1 to n do // loop over rows
3       for j = 1 to i-1 do // loop over columns
4           b_i := b_i - a_ij * x_j;
5       end for
6       x_i := b_i / a_ii;
7   end for

```

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 $x = [x_i]$ 1 **begin**2 **for** $j = 1$ **to** n **do**3 $x_j := b_j / a_{jj}$;4 **for** $i = j+1$ **to** n **do**5 $b_i := b_i - a_{ij}x_j$;

Theorem 2.1.8 Computational Cost of Forward Substitution

Number of floating point operations (+, −, ×, /) 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.*

Remark. *Solving upper triangular system using backward substitution.*

2.2 GEPP and Matrix Factorization**Theorem 2.2.1 Gaussian Elimination**

In Gaussian Elimination, we are allowed to

1. Swap rows (exchange, pivot)
2. Add multiple of one row to another
3. Multiply row by non-zero scalar.

Remark. *We require the equation with the largest coefficient in magnitude at the top when doing the Gaussian Elimination. This is because we want to divide by large numbers instead of smaller ones (which will cause errors).*

Algorithm 3: General Structure of GEPP

```

1 begin
2   for all stages do
3     pivot;
4     eliminate;

```

At stage k , eliminate x_k , from rows $k + 1$ to n

```

1  for i = k+1:n
2      m(i,k) = a(i,k) / a(k,k); % find the multiplier
3      a(i,k) = 0;
4      for j = k+1:n
5          a(i,j) = a(i,j) - m(i,k) * a(k,j); % could use vectorization
6      end
7      b(i) = b(i) - m(i,k) * b(k);
8  end

```

Pivoting at stage k : find the coefficient with the largest magnitude

```

1  %% The code tells us which row has the pivot.
2  p = k;
3  for i = k+1:n
4      if abs(a(p,k)) < abs(a(i,k))
5          p = i;
6      end
7  end
8  %% Swap rows in A and b
9  A([p,k], :) = A([k,p], :);
10 b(p) = b(k);

```

Theorem 2.2.2 Cost of GEPP

At stage k , we only focus on rows k through n and columns k through n . We have $(n - k)$ divisions for multipliers. For every multiplier we have $(n - k)$ multiplications, which are then used $(n - k)$ times to change each row. So, we have $(n - k)$ multiplications in total. Subtractions come with multiplications, so we also have $(n - k)(n - k)$ subtractions.

Theorem 2.2.3 Another Perspective on GEPP

The process of GEPP can be written as matrix multiplication EA , where E is an elementary matrix and act on the rows of A .

Theorem 2.2.4 Matrix Factorizations: $PA = LU$

$$PA = LU,$$

where U is upper-triangular, L is lower-triangular, and P is the pivot or permutation matrix.

This factorization comes from GEPP. Almost all matrices A have $PA = LU$ unless we have a column of all zeros in A .

Theorem 2.2.5 Solving $Ax = b$ with $PA = LU$

Given $Ax = b$ and pre-computed $PA = LU$:

$$PA = Pb$$

$$LU = Pb$$

$$Ux = L^{-1}Pb \quad \text{using forward substitution}$$

$$x = U^{-1}L^{-1}Pb \quad \text{using backward substitution}$$

This process need around $\mathcal{O}(n^2)$ operations. $\mathcal{O}(\cdot)$ is the big-O notation, meaning the number is dominated by n^2 .

PA = LU in MATLAB

```
1 [L, U, P] = lu(A);
2 % P could be omitted sometimes.
```

Theorem 2.2.6 Cholesky Factorization

If A is symmetric ($A = A^T$) and positive definite (all eigenvalues are positive), then $A = R^T R$, where R is upper-triangular and R^T is lower-triangular. This factorization is 2 time less expensive than GEPP.

Remark. Choleksy Factorization is just the $PA = LU$ factorization for SPDs (symmetric positive definite matrices).

Theorem 2.2.7 Other Matrix Factorization

1. QR decomposition:

$$A = QR,$$

where R is upper-triangular and Q is orthogonal such that $Q^T Q = Q^T Q = I$.

Q is really easy to be inverted and comes from the Gram-Schmidt process.

This composition is a bit more expensive than $PA = LU$.

2. Singular Value Decomposition (SVD):

$$A = U \Sigma V^T,$$

where U and V^T are orthogonal and Σ is diagonal. This factorization is also more expensive than $PA = LU$ decomposition.

2.3 Measuring Accuracy of Solutions

Definition 2.3.1 (Vector Norms). A vector norm is a function $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ that satisfies

- $\|x\| \geq 0 \quad \forall x \in \mathbb{R}^n$ and $\|x\| = 0$ if and only if $x = 0$.
- $\|cx\| = |c|\|x\| \quad \forall x \in \mathbb{R}^n$ and $c \in \mathbb{R}$.
- Triangular inequality: $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbb{R}^n$.

Definition 2.3.2 (Common Definitions of Norm).

- Pythagorean Distance: $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.
- Taxicab/Manhattan Distance: $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$.
- Infinity Norm: $\|x\|_\infty = \max_{i=1, \dots, n} |x_i|$.

Definition 2.3.3 (Measuring Errors). Suppose x is the true solution, and \hat{x} is the approximate solution. Then

$$\begin{aligned} \text{Error} &= \|\hat{x} - x\| \\ \text{Relative Error} &= \frac{\|\hat{x} - x\|}{\|x\|}, \quad x \neq 0. \end{aligned}$$

Remark. In practice, we do not know x , the true solution. So this measurement cannot be used.

Definition 2.3.4 (Residual). We know A , b , \hat{x} , and we want to solve $Ax = b$. So, the

$$\begin{aligned} \text{Residual} &= A\hat{x} - b \\ \text{Residual Norm} &= \|A\hat{x} - b\| \\ \text{Relative Residual Norm} &= \frac{\|A\hat{x} - b\|}{\|b\|} \end{aligned}$$