MA2213 Cheatsheet 19/20 Sem 1 Midterm

by Timothy Leong (format taken from Ning Yuan)

Lecture 2: Computer Arithmetic

Single-precision floating point format.

$$\underbrace{b_{31}}_{\text{sign}}\underbrace{b_{30}\cdots b_{23}}_{\text{8-bit exp}}\ \underbrace{b_{22}\cdots b_{0}}_{\text{23-bit mantissa}}$$

represents the binary number

$$(-1)^{b_{31}} \times 10^{b_{30}b_{29}\cdots b_{23}b_{23}-11111111} \times 1.b_{22}\cdots b_1$$

Example: Converting from decimal to IEEE.

$$(9.15625)_{10} = (1001.00101)_2 = \left(1.00100101 \times 10^{11}\right)_2$$

The mantissa is 00100101000000000000000 and the exponent is

$$(11 + 1111111)_2 = (10000010)_2$$

Note: The exponent is in the excess-127 format

Exceptions to the IEEE format.

Input	Output
0	00000000000000000000000000000000
-0	10000000000000000000000000000000
inf	01111111100000000000000000000000
-inf	11111111100000000000000000000000
nan	01111111110000000000000000000000
-nan	11111111110000000000000000000000

In fact, when the exponent is 11111111 and the mantissa is not zero, the result is always interpreted as NaN or -NaN, which means "Not-a-Number".

Denormal Numbers. If exponent is all zeros, the binary expression for denormal numbers is

$$(-1)^{b_{31}} \times 10^{-1111110} \times 0.b_{22}b_{21}\cdots b_1 b_0$$

and the corresponding decimal number is

$$(-1)^{b_{31}} \times 2^{-126} \times \sum_{i=1}^{23} b_{23-i}2^{-i}$$

Lecture 4: Linear Systems & Cramer's Rule

Matrix-vector multiplication.

Algorithm 1 Compute $\mathbf{y} = A\mathbf{x}$ given $A = (a_{ij})_{n\times n}$ and $x = (x_1, x_2, \cdots, x_n)^T$

for $i = 1, \cdots, n$ **do**
  $y_i \leftarrow 0$
end for
for $i = 1, \cdots, n$ **do**
  for $j = 1, \cdots, n$ **do**
    $y_i \leftarrow y_i + a_{ij}x_j$
  end for
end for
return $(y_1, y_2, \cdots, y_n)^T$

Time complexity: $O(n^2)$

Cramer's rule. Suppose A is an $n \times n$ matrix with non-zero determinant. Then the unique solution of $A\mathbf{x} = \mathbf{b}$ is given by

$$x_i = \frac{\det A_i}{\det A}, \quad i = 1, \cdots, n$$

where A_t is the matrix performed by replacing the ith column of A by the column vector b. **Time complexity:** $O(n^3)$

Lecture 5: Gaussian Elimination

Gaussian Elimination with no pivoting. The first 8 lines are elimination steps, and lines 9-16 are for backward substitution.

Algorithm 1 Solve the linear system with augmented matrix $\tilde{A} = (a_{ij})_{n\times(n+1)}$

1: **for** $j = 1, \cdots, n-1$ **do**
2:   **for** $i = j+1, \cdots, n$ **do**
3:     $m_{ij} \leftarrow a_{ij}/a_{jj}$
4:     **for** $k = j+1, \cdots, n+1$ **do**
5:       $a_{ik} \leftarrow a_{ik} - m_{ij}a_{jk}$
6:     **end for**
7:   **end for**
8: **end for**
9: $x_n \leftarrow a_{n,n+1}/a_{nn}$
10: **for** $i = n-1, \cdots, 1$ **do**
11:   $x_i \leftarrow a_{i,n+1}$
12:   **for** $j = i+1, \cdots, n$ **do**
13:     $x_i \leftarrow x_i - a_{ij}x_j$
14:   **end for**
15:   $x_i \leftarrow x_i/a_{ii}$
16: **end for**
17: **return** $(x_1, \cdots, x_n)^T$

Time complexity: $O(n^3)$

Gaussian Elimination with partial pivoting (naive).

Algorithm 2 Solve the linear system with augmented matrix $\tilde{A} = (a_{ij})_{n\times(n+1)}$

1: **for** $i = 1, \cdots, n$ **do**
2:   $r_i \leftarrow i$
3: **end for**
4: **for** $j = 1, \cdots, n-1$ **do**
5:   **if** $a_{r_j j} = 0$ **then**
6:     $k \leftarrow j+1$
7:     **while** $k \leqslant n$ and $a_{r_k j} = 0$ **do**
8:       $k \leftarrow k+1$
9:     **end while**
10:     **if** $k = n+1$ **then**
11:       **return** "Error: matrix is singular"
12:     **else**
13:       Swap r_j and r_k
14:     **end if**
15:   **end if**
16:   **for** $i = j+1, \cdots, n$ **do**
17:     $m_{ij} \leftarrow a_{r_i j}/a_{r_j j}$
18:     **for** $k = j+1, \cdots, n+1$ **do**
19:       $a_{r_i k} \leftarrow a_{r_i k} - m_{ij}a_{r_j k}$
20:     **end for**
21:   **end for**
22: **end for**
23: $x_n \leftarrow a_{r_n,n+1}/a_{r_n n}$
24: **for** $i = n-1, \cdots, 1$ **do**
25:   $x_i \leftarrow a_{r_i,n+1}$
26:   **for** $j = i+1, \cdots, n$ **do**
27:     $x_i \leftarrow x_i - a_{r_i j}x_j$
28:   **end for**
29:   $x_i \leftarrow x_i/a_{r_i i}$
30: **end for**
31: **return** $(x_1, \cdots, x_n)^T$

This algorithm only performs a swap when $a_{r_k j} = 0$. However, imprecise subtraction can cause a zero pivot to be non-zero.

Lecture 6: Pivoting strategy

Gaussian Elimination with partial pivoting (more reliable). We replace lines 5-15 in the above algorithm with this block of code, setting the entry with the largest magnitude as the pivot. **Time complexity:** $\mathbf{O(n^3)}$.

5: $l \leftarrow j$
6: **for** $k = j+1, \cdots, n$ **do**
7:   **if** $|a_{r_k j}| > |a_{r_l j}|$ **then**
8:     $l \leftarrow k$
9:   **end if**
10: **end for**
11: **if** $a_{r_l j} = 0$ **then**
12:   **return** "Error: matrix is singular"
13: **else**
14:   Swap r_j and r_l
15: **end if**

Gaussian Elimination with scaled partial pivoting (even more reliable). First, we locate the largest-magnitude number in each row and set that number as the scaling factor for that row.

for $i = 1, \cdots, n$ **do**
  $s_i \leftarrow |a_{i1}|$
  for $j = 2, \cdots, n$ **do**
    if $|a_{ij}| > s_i$ **then**
      $s_i \leftarrow |a_{ij}|$
    end if
  end for
  if $s_i = 0$ **then**
    return "Error: matrix is singular"
  end if
end for

Then for every elimination step, the same scaling factor is used. The corresponding pseudo code is just to replace line 7 to line 9 by

7: **if** $|a_{r_k j}|/s_{r_k} > |a_{r_l j}|/s_{r_l}$ **then**
8:   $l \leftarrow k$
9: **end if**

Thus the number of extra comparison is reduced to $n(n-1)$. For the linear system (8), this simplified algorithm still gives exact solutions.

Lecture 7: Matrix Factorization

Types of row operations.

1. $E_i \leftarrow E_i - m_{ij}E_j$ (multiply the jth row by m_{ij} and subtract the result from the ith row), where $i > j$. This is equivalent to multiplying the augmented matrix by the following matrix on the left-hand side:

$$I - m_{ij}\mathrm{e}_i\mathrm{e}_j^T$$

where I is the identity matrix, and e_i is the unit vector $(0, \cdots, 0, 1, 0, \cdots, 0)^T$, whose the only nonzero entry is its ith component. When $i > j$, the elements of the matrix $I - m_{ij}\mathbf{e}_i\mathbf{e}_j^T$ can be described as follows:

- The matrix is lower-triangular;
- All its diagonal entries are equal to 1;
- All its off-diagonal entries are zero except that its (i, j) -element is $-m_{ij}$.

2. $E_i \leftrightarrow E_j$ (exchange the ith row and the jth row) which is equivalent to pre-multiplying

$$I-\left(\mathbf{e}_i-\mathbf{e}_j\right)\left(\mathbf{e}_i-\mathbf{e}_j\right)^T$$

Features:

- The matrix is symmetric;
- All its diagonal entries are equal to 1 except the ith and jth ones are equal to zero;
- All its off-diagonal entries are zero except that its (i,j)-element and (j,i)-element are equal to 1.

Theorem 1. Suppose $j \in \{1,\cdots,n-1\}$ and $i_1, i_2, \cdots, i_k$ are integers in $\{j+1,\cdots,n\}$. Then

$$\begin{aligned}&\left(I- m_{i1}j\mathbf{e}_{i_1}\mathbf{e}_j^T\right)\left(I-m_{i_2j}\mathbf{e}_{i_2}\mathbf{e}_j^T\right)\cdots\left(I-m_{i_kj}\mathbf{e}_{i_k}\mathbf{e}_j^T\right)\\ &=I-\left(m_{i_1j}\mathbf{e}_{i_1}+m_{i_2j}\mathbf{e}_{i_2}+\cdots+m_{i_kj}\mathbf{e}_{i_k}\right)\mathbf{e}_j^T\end{aligned}$$

. When $i_1, i_2, \cdots, i_k$ are distinct, the matrix

$$I-\left(m_{i_1j}\mathbf{e}_{i_1}+m_{i_2j}\mathbf{e}_{i_2}+\cdots+m_{i_k\mathbf{j}}\mathbf{e}_{i_k}\right)\mathbf{e}_j^T$$

has the following structure:

- All the diagonal entries are equal to 1
- The (i_s,j) -element is m_{i_is} for any $s=1,\cdots,k$
- All other entries are equal to zero.

Or pictorially,

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\1/3&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\-1/3&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\-2/3&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$
$$=\begin{pmatrix}1&0&0&0\\-2/3&1&0&0\\-1/3&0&1&0\\1/3&0&0&1\end{pmatrix}$$

If $A\mathrm{x}=\mathrm{b}$ can be solved by GE without pivoting, then the GE is equivalent to pre-multiplying the matrix A by $n-1$ lower triangular matrices, and each lower triangular matrix has the form $I-\mathbf{m}_j\mathbf{e}_j^T$, and the process is equivalent to $\left(I-\mathbf{m}_{n-1}\mathbf{e}_{n-1}^T\right)\cdots\left(I-\mathbf{m}_2\mathbf{e}_2^T\right)\left(I-\mathbf{m}_1\mathbf{e}_1^T\right)A=U$, where U is an upper triangular matrix.

Theorem 2. Suppose $j<n$. Let $\mathbf{m}_j$ be the n -dimensional column vector $(0,\cdots,0,m_{j+1,j},\cdots,m_{nj})^T$. Then $\left(I-\mathbf{m}_j\mathbf{e}_j^T\right)^{-1}=I+\mathbf{m}_j\mathbf{e}_j^T$, i.e.

$$\begin{pmatrix}1&0&0&0\\-2/3&1&0&0\\-1/3&0&1&0\\1/3&0&0&1\end{pmatrix}^{-1}=\begin{pmatrix}1&0&0&0\\2/3&1&0&0\\1/3&0&1&0\\-1/3&0&0&1\end{pmatrix}$$

Theorem 3. Suppose $i_1 \leqslant i_2 \leqslant \cdots \leqslant i_k < n$, and for any $j=1,\cdots,k$, the vector $\mathbf{m}_j$ is an n-dimensional column vector whose first i_j components are zero. Then $\left(I+\mathbf{m}_1\mathbf{e}_{i_1}^T\right)\left(I+\mathbf{m}_2\mathbf{e}_{i_2}^T\right)\cdots\left(I+\mathbf{m}_k\mathbf{e}_{i_k}^T\right)=$
$I+\mathbf{m}_1\mathbf{e}_{i_1}^T+\mathbf{m}_2\mathbf{e}_{i_2}^T+\cdots\mathbf{m}_k\mathbf{e}_{i_k}^T$

Example of Theorem 3.

$$\begin{pmatrix}1&0&0&0\\2/3&1&0&0\\1/3&0&1&0\\-1/3&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&4/5&1&0\\0&1&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&5&1\end{pmatrix}$$
$$=\begin{pmatrix}1&0&0&0\\2/3&1&0&0\\1/3&4/5&1&0\\-1/3&1&5&1\end{pmatrix}.$$

Theorem 4. If $Ax=b$ can be solved by GE without pivoting, A can be factorised into L and U. $Ax=b \leftrightarrow LUx=b$. We can solve $Ly=b$ by forward substitution and then $Ux=y$ by backward substitution.

Algorithm 1 Solve $L\mathbf{y}=\mathbf{b}$ for $L=(\ell_{ij})_{n\times n}$ and $\mathbf{b}=(b_1,\cdots,b_n)^T$. L is unit lower-triangular.

```
y_1 ← b_1
for i = 2, ..., n do
    y_i ← b_i
    for j = 1, ..., i-1 do
        y_i ← y_i - ℓ_ij y_j
    end for
end for
return (y_1, ..., y_n)^T
```

Algorithm 2 Solve $U\mathbf{x}=\mathbf{y}$ for $U=(u_{ij})_{n\times n}$ and $\mathbf{y}=(y_1,\cdots,y_n)^T$. U is upper-triangular.

```
x_n ← y_n/u_nn
for i = n-1, ..., 1 do
    x_i ← y_i
    for j = i+1, ..., n do
        x_i ← x_i - u_ij y_j
    end for
    x_i ← x_i/u_ii
end for
return (x_1, ..., x_n)^T
```

The time complexity of solving a linear system by LU-factorization is $O(n^2)$

Tutorials

Tut 1 Exercise 3.

$$B(x,y)=\sum_{n=0}^{+\infty}(-1)^n\frac{(y-1)\cdots(y-n)}{n!(x+n)}$$

Given x and y, write down the pseudocode to approximate the value of $B(x,y)$

Algorithm 1 Compute $B(x,y)$

```
B ← 0, S ← 1, n ← 0, w = 1/x
while |w| ⩾ ε do
    B ← B + w
    n ← n + 1
    S ← -((y-n)/n) S
    w ← S/(x+n).
end while
return B + w
```

Tut 2 Exercise 2. Let $A=(a_{ij})_{n\times n}$ be an upper-triangular matrix and $B=(b_{ij})_{n\times n}$ be a lower- triangular matrix. Write an algorithm to compute $C=AB$ and count the number of arithmetic operations.

Algorithm 1 Compute $C=AB$ with A being upper-triangular and B being lower-triangular

```
for i = 1, ..., n do
    for j = 1, ..., n do
        c_ij ← 0
        for k = max(i, j), ..., n do
            c_ij ← c_ij + a_ik b_kj
        end for
    end for
end for
```

The number of multiplications/additions is

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=i}^{n}1+\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j}^{n}1=\frac{1}{6}n(n+1)(2n+1).$$

Tut 2 Exercise 3. Write the pseudocode that uses GE without pivoting to solve the linear system $A\mathrm{x}=\mathrm{b}$ with $\mathrm{b}=(b_1,\cdots,b_n)^T$ and A being an upper-Hessenberg matrix

$$A=\begin{pmatrix}a_{11}&a_{12}&a_{13}&\cdots&a_{1,n-1}&a_{1n}\\a_{21}&a_{22}&a_{23}&\cdots&a_{2,n-1}&a_{2n}\\&a_{32}&a_{33}&\cdots&a_{3,n-1}&a_{3n}\\&&a_{43}&\cdots&a_{4,n-1}&a_{4n}\\&&&\ddots&\vdots&\vdots\\0&&&&a_{n,n-1}&a_{nn}\end{pmatrix}$$

Algorithm 2 Solve $A\mathbf{x}=\mathbf{b}$ for an upper-Hessenberg matrix A

```
for i = 1, ..., n-1 do
    m ← a_{i+1,i}/a_ii
    for j = i+1, ..., n do
        a_{i+1,j} ← a_{i+1,j} - m a_ij
    end for
    b_{i+1} ← b_{i+1} - m b_i
end for
x_n ← b_n/a_nn
for i = n-1, ..., 1 do
    x_i ← b_i
    for j = i+1, ..., n do
        x_i ← x_i - a_ij x_j
    end for
    x_i ← x_i/a_ii
end for
return (x_1, ..., x_n)^T
```

The number of subtractions/multiplications is n^2-1; the number of divisions is $2n-1$. The idea is to only do a row subtraction with the row immediately below.

Miscellaneous

Summation formulae.

$$\sum_{k=1}^{n}k=\frac{n(n+1)}{2}$$
$$\sum_{k=1}^{n}k^2=\frac{n(n+1)(2n+1)}{6}$$
$$\sum_{k=1}^{n}k^3=\frac{n^2(n+1)^2}{4}$$
$$\sum_{k=1}^{n}k(k+1)=\frac{n(n+1)(n+2)}{3}$$
$$\sum_{k=1}^{n}(2k-1)=n^2$$

Also from tutorial 2

$$\sum_{i=1}^{n}\left(\left(n+\frac{1}{2}i\right)-\frac{1}{2}i^2\right)=\frac{n(n+1)(2n+1)}{6}$$