ECE 5340/6340 Lecture 5: Gaussian Elimination

METHODS OF SOLVING MATRIX EQUATIONS:

a) Direct
   - Gaussian Elimination <<< We will study
   - LU Decomposition

b) Iterative
   - SOR: Successive Over-Relaxation <<
   - Conjugate Gradient Method

GAUSSIAN ELIMINATION

Example: Solve

$$\begin{aligned} 3x_1 + 2x_2 + 4x_3 &= 19 \\ 2x_1 + 6x_2 + 5x_3 &= 29 \\ x_1 + x_2 + x_3 &= 6 \end{aligned}$$

Exact solution: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$

1) Write as an augmented matrix: A|b

$$\begin{array}{ccc|c} 3 & 2 & 4 & 19 \\ 2 & 6 & 5 & 29 \\ 1 & 1 & 1 & 6 \end{array}$$

2) Eliminate all but x_1 from first equation:

$$\begin{array}{ccc|c} (3 & 2 & 4 & 19\ )* -2/3 \\ +(2 & 6 & 5 & 29\ ) \end{array}$$

$$
\begin{array}{llll}
0 & 14/3 & 7/3 & \mid 49/3 \quad << \text{New second row}
\end{array}
$$

$$
\begin{array}{llll}
(3 & 2 & 4 & \mid \quad 19 \;)* -1/3 \\
+(1 & 1 & 1 & \mid \quad 6 \;) \\
\hline
0 & 1/3 & -1/3 & \mid \quad -1/3 \quad << \text{New third row}
\end{array}
$$

$$
\begin{array}{llll}
3 & 2 & 4 & \mid \quad 19 \quad << \text{New matrix} \\
0 & 14/3 & 7/3 & \mid 49/3 \\
0 & 1/3 & -1/3 & \mid \quad -1/3
\end{array}
$$

3) Eliminate x_2 from all but second row:

$$
\begin{array}{llll}
(0 & 14/3 & 7/3 & \mid 49/3) * (-14/3)/(1/3) \\
+ (0 & 1/3 & -1/3 & \mid \quad -1/3) \\
\hline
0 & 0 & -1/2 & \mid \quad -3/2 \quad << \text{New third row}
\end{array}
$$

$$
\begin{array}{llll}
3 & 2 & 4 & \mid \quad 19 \quad << \text{New matrix} \\
0 & 14/3 & 7/3 & \mid 49/3 \\
0 & 0 & -1/2 & \mid \quad -3/2
\end{array}
$$

4) Solve using Back Substitution starting at bottom:

$-1/2\ x_3 = -3/2 \rightarrow x_3 = 3$

$14/3\ x_2 + 7/3\ x_3 = 49/3 \rightarrow x_2 = 2$

$3\ x_1 + 2\ x_2 + 4\ x_3 = 19 \rightarrow x_1 = 1$

5) Check your solution in original matrix:

$$
\begin{array}{llll}
\mid 3 & 2 & 4 \mid & \mid 1 \mid = \mid 19 \mid \\
\mid 2 & 6 & 5 \mid & \mid 2 \mid \quad \mid 29 \mid
\end{array}
$$

| 1 1 1 | |6| | 6 |

WHAT CAN GO WRONG (Besides programming error)

- Pivoting
- Scaling
- Ill-conditioning

NUMERICAL ERROR and Floating point arithmetic:

Reading (Handout)

Computer represents real numbers by truncated real numbers. This causes numerical errors, which are generally small. IF, however, you are working in a range where these errors are significant, then they can cause problems.

Example: Add 1/3 + 1/3 + 1/3 using 4 bit floating point arithmetic. (Computers are generally 8- or 16-bit in single precision). Exact answer: 1

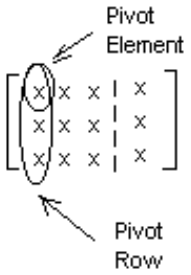
$$\begin{array}{r} 0.3333 = a \\ 0.3333 = b \\ +\ 0.3333 = c \\ \hline 0.9999 \end{array}$$

Now, suppose you had used this sum as a variable in your program… IF ((a+b+c) = 1) then_do_something, your program will malfunction because of the floating point arithmetic. 0.9999 does NOT equal integer 1. It is ZERO!

Computers TRUNCATE. They do not round.

Many numerical errors are CUMULATIVE (they get worse as the problem is larger, or as an iterative solution is allowed to run longer.)

PIVOTING:



The pivot element should be the largest MAGNITUDE element in the row.

Example: Solve (upside down step 2 above)

0	1/3	-1/3	|	-1/3
0	14/3	7/3	|	49/3
3	2	4	|	19

TO reduce row 3, you would multiply row 1 by 3/0, which blows up. Pivoting AT EACH STEP will prevent this. If element is not zero, but is small, numerical error increases.

PARTIAL PIVOTING algorithm:
Interchange rows until largest magnitude element is the pivot element.

3	2	4	|	19	<<< This is OK for step 1.
0	1/3	-1/3	|	-1/3	
0	14/3	7/3	|	49/3	

Now pivot for step 2:

3	2	4	|	19
0	***14/3***	7/3	|	49/3

0 1/3 -1/3 | -1/3

SCALING:

Recall that a matrix represents a set of vectors. If one of the vectors is significantly longer than the others, this does not affect the solution IN THEORY, but it causes numerical errors which affect the solution in practice.

Example: Solve this with 4-bit floating point arithmeetic:

$$x_1 + 10^{12} x_2 = 1 + 10^{12}$$
$$x_1 - x_2 = 0$$

Answer: $x_1 = x_2 = 1$

Reduce:

$$x_1 + 10^{12} x_2 = 1 + 10^{12} = 1000000000001$$

$$x_1 + 10^{12} x_2 = 10^{12} \ll \text{(4 decimal places accurate)}$$
$$-(x_1 - x_2 = 0)$$
$$10^{12} x_2 = 10^{12} \ll \text{(4 decimal places accurate)}$$

$$\rightarrow x_2 = 1$$

Substitute into second equation: $x_1 = 1$

Substitute into first equation (just as legit): $x_1 = 0$

This solution is flakey!

How do you tell when a solution is scaled poorly, and what do you do about it?

The b vector should all be similar order of magnitude. One easy way of getting this is to divide each equation through by its bi value. Then the b vector = 1.

PIVOTING ALGORITHM

At each step of gaussian elimination, put the largest element in the column on the diagonal.

```
DO  L = 1,M-1                                   ! which step of the elimination you are on

c --- Find pivot element and location --
        pivot = 0                               ! initialize pivot element
        ipivot = 0                              ! initialize pivot row location
        DO I = L , M                            ! find pivot element
                IF ( | a(I,L) | > pivot) THEN
                        pivot = a(I,L)          ! store new pivot element
                        ipivot = I              ! store location of new pivot element
                ENDIF
        ENDDO

c --- Exchange Lth row and pivot row to put pivot element on top ---
        DO J = L , N                            ! for each non-zero element in the row
                holder = a(L,J)                 ! hold the value currently in the top row
                a(L,J) = a(ipivot,J)            ! move the element in ipivot row to top row
                a(ipivot,J) = holder            ! put top row element into ipivot row
        ENDDO

ENDDO
```

SCALING ALGORITHM (One of many methods)

Before your start elimination, find the magnitude of each vector (row in the array), and make it a unit vector. This makes all the vectors the same size.

```
DO I = 1 , M                                    ! for Each row

        C – Find length of each vector (matrix row)
        DO J = 1 , N
                vector_length = vector_length +  a(I ,J ) 2
        ENDDO
        vector_length = sqrt(vector_length)

        --Scale each vector  to a unit length (=1.0) –
```

```
DO J = 1 , N
        a(I , J ) = a(I , J) / vector_length
ENDDO

ENDDO
```