Math 471 (Numerical methods)

Chapter 3 (first half). System of equations

Overlap §3.1–3.4 of Bradie

§3.0 Introduction and Review of Linear Algebra.

System of equations have more than one unknowns and equations.

• Example. Find the intersections of the unit circle $x^2+y^2=1$ and the sine function $y=\sin(x)$. We write it as

$$\begin{cases} x^2+y^2=1 \\ y=\sin(x). \end{cases}$$

This is a nonlinear system and can be numerically solved using the Newton's method — which will be discussed later.

A more basic type of system is linear system where the unknowns appear linearly (with power 1)

$$\begin{cases} a_{11}x_1+a_{12}x_2+...+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+...+a_{2n}x_n=b_1 \\ \qquad\qquad\qquad\qquad\qquad\qquad ..... \\ a_{m1}x_1+a_{m2}x_2+...+a_{mn}x_n=b_m \end{cases}$$

Here a_{ij}, b_i are constants. It is a system of m equations with n unknowns.

Note. n and m can be unequal, which is important sometimes!

Alternatively, matrix notation

$$Ax=\vec{b},$$

where the coefficient matrix is

$$A=\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix}$$

Now, set $n=m$ so A becomes a square matrix and try to solve

$$Ax=\vec{b}.$$

Matlab code `x=A\b` will simply do but may be slow or inaccurate. Plus, the goal of this course is to look under the hood and study how such algorithm is implemented in detail.

Matrix multiplication. It is ok to multiple two matrices AB only of their "inner dimensions" agree, that is, A is n-by-m and B is m-by-k. As a result, the (a, b) entry of AB equals the dot product of the a-th row of A and b-th column of B. We sometimes use the following notations

$$(AB)_{ab} = \sum_{i=1}^{m} (A)_{ai}(B)_{ib}.$$

Important. Matrix multiplication is in general not commutatble, $AB \neq BA$, except in very special circumstances.

Notations.

1. Identity matrix I or I_n with n indicating it is an n-by-n matrix. It has 1 on diagonal and 0 otherwise. Always, given an n-by-m matrix M, we have $I_n M = M I_m = M$. (Notice the dimensions of I)
2. Inverse matrix. Given n-by-n matrix A, its inverse A^{-1} is such that

$$AA^{-1} = A^{-1}A = I.$$

Not every matrix is invertible. When it is, the solution to $A\vec{x} = \vec{b}$ is simply

$$\vec{x} = A^{-1}\vec{b}.$$

§3.1 Gaussian Elimination for Solving $A\vec{x} = \vec{b}$.

Let's practice Gaussian elimination on paper and then implement it as an algorithm.

But first, note that Gaussian elimination works for both $A\vec{x} = \vec{b}$ and $AX = I$, the latter of which essentially inverts A since $X = A^{-1}$.

• Example. Solve for x_1, x_2, x_3 in the system of linear equations

$$\begin{cases} 2x_1 - x_2 = -1 \\ -x_1 + 2x_2 - x_3 = 1 \\ -x_2 + 2x_3 = 1 \end{cases}$$

Write coefficient matrix A and the RHS vector $\vec{b}$ together to form the augmented matrix.

$$(A|\vec{b}) = \left(\begin{array}{ccc|c} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 1 \\ 0 & -1 & 2 & 1 \end{array}\right).$$

Then, perform elementary row operations[1] to reduce A to a upper triangular matrix.

$$\text{(cont. from above) ... subtract row (1) } \times -\frac{1}{2} \text{ from row (2)} \begin{pmatrix} 2 & -1 & 0 & | & -1 \\ 0 & 3/2 & -1 & | & 1/2 \\ 0 & -1 & 2 & | & 1 \end{pmatrix}$$

$$\text{... subtract row (2) } \times -\frac{2}{3} \text{ from row (3)} \begin{pmatrix} 2 & -1 & 0 & | & -1 \\ 0 & 3/2 & -1 & | & 1/2 \\ 0 & 0 & 4/3 & | & 4/3 \end{pmatrix}$$

Here, we "sweep" through entries below the diagnonal in the order of: 1st column → 2nd column → 3th column → ... such an order is necessary to guarantee the zeroing of each column does NOT affect the zeros in preceeding columns to its left.

In the next step, having an uppper triangular matrix makes it possible to do backward substitution which is essentially a series of row operations. The goal of backward substitution is to reduce A further to a diagnonal matrix (preferrably an Identity matrix). Thus, we need to "sweep" through entries above the diagonal in the reverse order of: nth column → $n-1$th column → ... such an order is necessary to guarantee the zeroing of each column does NOT affect the zeros in preceeding columns to its right.

$$\text{(cont. from above) ... row (3) } \times \frac{3}{4} \begin{pmatrix} 2 & -1 & 0 & | & -1 \\ 0 & 3/2 & -1 & | & 1/2 \\ 0 & 0 & 1 & | & 1 \end{pmatrix}$$

$$\text{... subtract row (3) } \times -1 \text{ from row (2)} \begin{pmatrix} 2 & -1 & 0 & | & -1 \\ 0 & 3/2 & 0 & | & 3/2 \\ 0 & 0 & 1 & | & 1 \end{pmatrix}$$

$$\text{... row (2) } \times \frac{2}{3} \begin{pmatrix} 2 & -1 & 0 & | & -1 \\ 0 & 1 & 0 & | & 1 \\ 0 & 0 & 1 & | & 1 \end{pmatrix}$$

$$\text{... subtract row (2) } \times -1 \text{ from row (1)} \begin{pmatrix} 2 & 0 & 0 & | & 0 \\ 0 & 1 & 0 & | & 1 \\ 0 & 0 & 1 & | & 1 \end{pmatrix}$$

[1]The 3 type of row operations include: 1. subtract a nonzero multiple of one row from another row; 2. mupliply a row by a nonzero number; 3. switch two rows.

$$\ldots \text{ row } (1) \times \frac{1}{2} \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

So the answer is

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

During these steps, the diagonal entries are called **pivots**. Each row operation always involve rescaling of a pivot to match the off-diagonal entries being zeroed.

Now, if we do the same sequence of row operations to the augmented matrix

$$(A|I) = \left(\begin{array}{ccc|ccc} 2 & -1 & 0 & 1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 2 & 0 & 0 & 1 \end{array}\right),$$

then we will arrive at

$$(I|A^{-1}) = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 3/4 & 1/2 & 1/4 \\ 0 & 1 & 0 & 1/2 & 1 & 1/2 \\ 0 & 0 & 1 & 1/4 & 1/2 & 3/4 \end{array}\right).$$

In general, under row operations (first row reduction on A to lower triangular matrix, then back substitution on A to finish at identity matrix)

$$(A|B) \ \rightarrow \ \ldots \ \rightarrow \ (I|A^{-1}B)$$

Matlab code

```
function X=main(A,B) % solve for X in AX=B using the Guassian Elimination


% Part I .... reduce A to upper triag.
n=size(A);
Aug=[A B]; % create the augmented matrix
for col=1:n % sweep the columns from the left
for row=col+1:n % zero the off diag entry at (row,col)
                    % below the pivot at (col,col)
       k=Aug(row,col)/Aug(col,col);
       Aug(row,:)=Aug(row,:)-k*Aug(col,:); % row operation
```

```
    end
end

% Part II ... further reduce A to identity.
% But first, normalize all the diagonal entries to one
  for row=1:n
     Aug(row,:)=Aug(row,:)/Aug(row,row); % row operation
  end

  for col=n:-1:1 % sweep the columns from the right
    for row=col-1:-1:1 % zero the off diag entry at (row,col)
                       % above the pivot at (col,col)
       k=Aug(row,col); % effectively k=Aug(row,col)/Aug(col,col)
                       % with Aug(col,col)=1
       Aug(row,:)=Aug(row,:)-k*Aug(col,:); % row operation
    end
end
X=Aug(:,n+1:end); % the solution is column n+1 of Aug through the last column.
```

Operation count In part I, the two "for" loops generates

$$(n-1)+(n-2)+...+1=\frac{n(n-1)}{2} \text{ iterations.}$$

In each iteration, `Aug(row,:)=Aug(row,:)-k*Aug(col,:);` is the main part and uses n multiplications and n subtractions. Therefore, the total operation count for part I is $n\cdot\frac{n(n-1)}{2}$ multiplications and $n\cdot\frac{n(n-1)}{2}$ subtractions, i.e. totally

$$n\cdot n\cdot(n-1) \text{ operations.}$$

We will use symbol $O(n^3)$ to denote the operation complexity of this part of algorithm. Such a symbol emphasize on the degree/order of the operation count as a 3rd order polynomial while omitting the much less important constant coeffiecient and lower order terms.

Likewise, the operation count of part II is also $O(n^3)$.

§3.2 Pivoting — a conditioning strategy.

Pivoting, as a verb, denotes a family of strategies that switch entries in and out of pivot position using row/column exchange. The resulting matrix tends to have larger pivot elements on the diagonal, which helps deal with round-off error.

Reason: Gaussian elimination breaks down if a pivot element is zero. In general, the pivot element can be very small (but nonzero) and still has the undesired effect of amplifying errors.

• Example. The system $A\vec{x} = \vec{b}$ with

$$A = \begin{pmatrix} 0 & 4 & -15 \\ 10 & 0 & 15 \\ 1 & -1 & -1 \end{pmatrix} \text{ and } \vec{b} = \begin{pmatrix} -12 \\ 100 \\ 0 \end{pmatrix}.$$

The pivot on entry (1,1) is zero, making it impossible to even start row operations!

Remedy. Switch row (1) and (2). Or, switch row (1) and (3). The former is preferred since we like the pivot to be away from zero as far as possible.

Another remedy. Switch *column* (1) and (3). Then, the position of x_1 and x_3 is switched as well. Need some bookkeeping.

• Example. The system $A\vec{x} = \vec{b}$ with

$$A = \begin{pmatrix} \varepsilon & 1 & | & 1+\varepsilon \\ 1 & 1 & | & 2 \end{pmatrix} \text{ and } \vec{b} = \begin{pmatrix} 1+\varepsilon \\ 2 \end{pmatrix}.$$

The exact solution is $\vec{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

The pivots are nonzero and Guassian elimination seems to work. It does work if the digits are all kept. But in finite precision arithmetic, ε is so small that $1 + \varepsilon$ is rounded off as 1 (but ε is NOT rounded off to 0. why?). Thus the augmented matrix is stored by the computer as

$$\begin{pmatrix} \varepsilon & 1 & | & 1 \\ 1 & 1 & | & 2 \end{pmatrix} \rightarrow \text{ row operation}$$

$$\rightarrow \begin{pmatrix} \varepsilon & 1 & | & 1 \\ 0 & 1-\frac{1}{\varepsilon} & | & 2-\frac{1}{\varepsilon} \end{pmatrix}$$

Again, due to round-off error, the above matrix is stored by the computer as

$$\begin{pmatrix} \varepsilon & 1 & | & 1 \\ 0 & -\frac{1}{\varepsilon} & | & -\frac{1}{\varepsilon} \end{pmatrix} \rightarrow \text{ row operations ...}$$
$$\rightarrow \begin{pmatrix} 1 & 0 & | & 0 \\ 0 & 1 & | & 1 \end{pmatrix}$$

(0,1) is too far away from the exact solution (1,1) !

Remedy. The first pivot is too small so we exchange row (1) and row (2),

$$\begin{pmatrix} 1 & 1 & | & 2 \\ \varepsilon & 1 & | & 1 \end{pmatrix} \rightarrow \text{ row operation}$$
$$\rightarrow \begin{pmatrix} 1 & 1 & | & 2 \\ 0 & 1-\varepsilon & | & 1-2\varepsilon \end{pmatrix} \rightarrow \text{ round-off}$$
$$\rightarrow \begin{pmatrix} 1 & 1 & | & 2 \\ 0 & 1 & | & 1 \end{pmatrix} \rightarrow \text{ row operations ...}$$
$$\rightarrow \begin{pmatrix} 1 & 0 & | & 1 \\ 0 & 1 & | & 1 \end{pmatrix} \text{ All right!}$$

Another remedy. Switch the column.

Algorithm: partial pivoting. Interchange **rows** to have the largest possible entry in absolute value at the pivot position. That is, each time before reducing (the lower part of) the pivot column $j = col$ to zeros, find the largest entry in absolute value from the lower part of column $j = col$,

$$\begin{pmatrix} a_{j,j} \\ a_{j+1,j} \\ ... \\ a_{n,j} \end{pmatrix}$$

Let $a_{k,j}$ be the candidate. Then, exchange row (j) and row (k) so that, the pivot element now has the largest absolute value among the lower part of column $j = col$.

The first part of Guassian elimination needs to be modified. In the following code, `findmax` and `row_op1` are user-defined subroutines.

```
for col=1:n % sweep the columns from the left
    m=findmax(Aug,col); % this subroutine returns the index of the largest
```

```
                        % abs value from all entries below diag in column col
    Aug=row_op1(Aug,m,col); % exchange row m and col
    for row=col+1:n % zero the off diag entry at (row,col)
                    % below the pivot at (col,col)
       k=Aug(row,col)/Aug(col,col);
       Aug(row,:)=Aug(row,:)-k*Aug(col,:); % row operation
    end
end
```

§3.3 Vector and matrix norms.

Motivation: how do we measure and compare the errors in computing numerical solutions of linear systems? For instance, if we have two approximate solutions $\vec{y}$ and $\vec{z}$ to the linear system $A\vec{x} = \vec{b}$, their errors are given by

$$\vec{e}_y = \vec{x} - \vec{y}, \quad \vec{e}_z = \vec{x} - \vec{z}.$$

How do we compare these two vectors? Sometimes, the residuals are compared

$$\vec{r}_y = \vec{b} - A\vec{y}, \quad \vec{r}_z = \vec{b} - A\vec{z}.$$

Again, they are both vectors.

Therefore, it is necessary to assign each vector a **scalar** number, which is called "norm".

Definition. Given a vector with n components $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}$, its norm $\|\vec{x}\|$ is any[2] scalar-valued function satisfying

1. *Positivity*: $\|\vec{x}\| \geq 0$. $\|\vec{x}\| = 0$ if and only if $\vec{x} = \vec{0} = \Big(0, 0, ..., 0\Big)^T$. (Counterexample: For $n > 1$, $\|\vec{x}\| = x_1^2$ is NOT a norm. Why?)

2. *Scalability*: $\|\alpha\vec{x}\| = |\alpha| \cdot \|\vec{x}\|$ for any scalar number α. Notice the difference in the notations of $|\cdot|$ and $\|\cdot\|$. The former is for scalar and the latter vector.

[2] therefore, multiple choices for norm

3. *Triangle inequality*: $\|\vec{x}+\vec{y}\| \le \|\vec{x}\| + \|\vec{y}\|$.

• Example. The infinity norm $\|\vec{x}\|_\infty = \max_{i=1,2,\dots,n}\{|x_i|\}$.
(Optional) Why is it a valid norm? 1. positivity. 2. Scalability. 3. Triangle inequality.

$$\begin{aligned}\|\vec{x}+\vec{y}\|_\infty &= \max\{|x_i+y_i|\} \\ &\le \max\{|x_i|+|y_i|\} \text{ ... triangle inequality for absolute values} \\ &\le \max\{|x_i|\} + \max\{|y_i|\} = \|\vec{x}\|_\infty + \|\vec{y}\|_\infty\end{aligned}$$

• Example. The 2-norm

$$\|\vec{x}\|_2 = \sqrt{\vec{x}\cdot\vec{x}} = \sqrt{\vec{x}^T\vec{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

a.k.a. the Euclidean length of a vector (Geometry: Pythagorean theorem)
(Optional). Why a valid norm? Condition 1,2 are easy to verify. For the triangle inequality, however, one needs to use the Cauchy-Schwartz inequality

$$\sum x_i y_i \le \sqrt{\sum x_i^2}\sqrt{\sum y_i^2} \Leftrightarrow \vec{x}^T\vec{y} \le \|\vec{x}\|_2\|\vec{y}\|_2$$

so that

$$\begin{aligned}\|\vec{x}+\vec{y}\|_2 &= \sqrt{(\vec{x}+\vec{y})^T(\vec{x}+\vec{y})} \text{ ... by definition} \\ &= \sqrt{\vec{x}^T\vec{x} + \vec{x}^T\vec{y} + \vec{y}^T\vec{x} + \vec{(}y)^T\vec{(}y)} \\ &\le \sqrt{\|\vec{x}\|^2 + 2\|\vec{x}\|\|\vec{y}\| + \|\vec{y}\|^2} \text{ ... by Cauchy-Schwartz} \\ &= \|\vec{x}\| + \|\vec{y}\|\end{aligned}$$

• Example. Let $\vec{x} = \begin{pmatrix}2,-4,-1\end{pmatrix}^T$ and $\vec{y} = \begin{pmatrix}3,3,-2\end{pmatrix}^T$. Compare $\vec{x}$ and $\vec{y}$ in in terms of the infinity norm. What about the 2-norm?

Note. There are many other ways to define norms as long as the 3 conditions are satisfied. e.g. the 1-norm $\|\vec{x}\|_1 = \sum |x_i|$.

§3.4 Error analysis using norms

Given an approximate solution $\tilde{x}$ to $A\vec{x} = \vec{b}$, we have an exact system and an approximate system

$$A\vec{x} = \vec{b}, \quad A\tilde{x} = \tilde{b}.$$

Taking their difference, we have

$$A(\vec{x} - \tilde{x}) = \vec{b} - \tilde{b}.$$

Thus, **define** error and residual as

$$\vec{e} = \vec{x} - \tilde{x}, \quad \vec{r} = \vec{b} - \tilde{b}.$$

They are related as

$$\vec{r} = A\vec{e} \tag{1}$$

which implies, if A is invertible, then

$$\tilde{x} = \vec{x} \text{ iff } \vec{e} = 0 \text{ iff } \vec{r} = 0.$$

However, the following is not necessarily true

$$\|\vec{e}\| < \varepsilon \text{ iff } \|\vec{r}\| < \varepsilon.$$

In fact, the error $\vec{e}$ and residual $\vec{r}$, compared in norm, can differ in multiple magnitudes because of the multiplication of matrix A! e.g. $\vec{e} = \begin{pmatrix} 0.01 \\ 0.02 \end{pmatrix}$, $A = \begin{pmatrix} -1 & 100 \\ -2 & 0 \end{pmatrix} \rightarrow \vec{r} = A\vec{e} = \begin{pmatrix} 2 \\ -0.02 \end{pmatrix}$ which is about 100 times larger than $\vec{e}$ in terms of the infinity norm. (Exercise: How much difference if measured in the 2-norm?)

To this end, we need some measurement for the matrix A in terms of its "amplifying" ability. **Define** matrix norm for any matrix A **associated with** some given vector norm.

$$\|A\| = \max_{\vec{x} \neq 0} \frac{\|A\vec{x}\|}{\|\vec{x}\|}$$

- Example. the infinity norm of a matrix A is

$$\|A\|_\infty = \max_{\vec{x} \neq 0} \frac{\|A\vec{x}\|_\infty}{\|\vec{x}\|_\infty}$$

- Example. the 2-norm of a matrix A is

$$\|A\|_2 = \max_{\vec{x} \neq 0} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}$$

- Example. Use the previous example, $A = \begin{pmatrix} -1 & 100 \\ -2 & 0 \end{pmatrix}$. It's infinity norm is

$$\|A\|_\infty = 101,$$

i.e., given any vector $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, multiplying it with A will result in at most 101 times magnigication of the infinity norm.

$$\|A\vec{x}\|_\infty \leq 101\|\vec{x}\|_\infty.$$

In fact, setting $\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ we have $\|A\vec{x}\|_\infty = 101$ and $\|\vec{x}\|_\infty = 1$. Set $\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and we have $\|A\vec{x}\|_\infty = 2$ and $\|\vec{x}\|_\infty = 1$.

Matrix norm helps us to relate the error with residual since we have equation (1).

Theorem 1 *Given a vector norm, we can relate the error $\vec{e}$ and residual $\vec{r}$ both ways in terms of the associated matrix norms*

$$\|\vec{r}\| \leq \|A\|\|\vec{e}\| \quad \textit{and} \quad \|\vec{e}\| \leq \|A^{-1}\|\|\vec{r}\|.$$

Question: how do we effectively calculate the associated matrix norm?

Theorem 2

$$\|A\|_\infty = \max_i\{\|\vec{a}_i\|_1 = \sum_{j=1}^{n} |a_{ij}| \ : \ \vec{a}_1, \vec{a}_2, ..., \vec{a}_n \textit{ are the rows of } A\},$$

$$\|A\|_2 = \max\{\sqrt{\lambda_i} \ : \ \lambda_1, ...\lambda_n \textit{ are the eigenvalues of } A^TA\}.$$

Proof. (skipped)

Use the previous example $A = \begin{pmatrix} -1 & 100 \\ -2 & 0 \end{pmatrix}$. Then, we see why $\|A\|_\infty = 101$. Also, the 2-norm of A is the square root of the largest eigenvalue of $A^TA = \begin{pmatrix} 5 & -100 \\ -100 & 10^4 \end{pmatrix}$. (Exercise)

§3.4 Error analysis (cont'd).

Now, we are ready to look at one of the essential theorems in numerical linear algebra. The starting point is error analysis: when solving $A\vec{x} = \vec{b}$ in computer, loss of accuracy often occurs due to various reasons, e.g. round-off error. Then,

By having some residual $\vec{r}$ on the RHS term $\vec{b}$, how much error $\vec{e}$ do we expect on the solution $\vec{x}$?

One answer comes from Theorem 1 $\|\vec{e}\| \leq \|A^{-1}\| \, \|\vec{r}\|$.

However, we are more interested in **relative** error $\dfrac{\|\vec{e}\|}{\|\vec{x}\|}$ and **relative** residual $\dfrac{\|\vec{r}\|}{\|\vec{b}\|}$. These are the quantities that affect the number of correct significant digits. (Why?)

Theorem 3

$$\frac{\|\vec{e}\|}{\|\vec{x}\|} \leq \kappa(A)\frac{\|\vec{r}\|}{\|\vec{b}\|}, \text{ where } \kappa(A) = \|A\|\,\|A^{-1}\|, \text{ called } \underline{\text{condition number}}.$$

Proof. Use relations $\vec{b} = A\vec{x} \Rightarrow \|\vec{b}\| \leq \|A\|\,\|\vec{x}\|$ and $\vec{e} = A^{-1}\vec{r} \Rightarrow \|\vec{e}\| \leq \|A^{-1}\|\,\|\vec{r}\|$. Then the conclusion follows from multiplying these two inequalities.

• Example. Take $A = \begin{pmatrix} 101 & 99 \\ 99 & 101 \end{pmatrix}$ and $\vec{b} = \begin{pmatrix} 200 \\ 200 \end{pmatrix}$ so that $\vec{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. By introducing residual $\vec{r} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$ to the RHS $\vec{b}$, we end up with error $\vec{e} = A^{-1}\vec{r} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$.So, the relative error and residual in terms of infinity norm are

$$\frac{\|\vec{e}\|_\infty}{\|\vec{x}\|_\infty} = 1$$

$$\frac{\|\vec{r}\|_\infty}{\|\vec{b}\|_\infty} = 0.01$$

A 100 times amplification from relative residual to relative error.

Check: the condition number of A is

$$\kappa(A) = \|A\|_\infty\,\|A^{-1}\|_\infty = 100.$$

(Exercise: compute $\kappa(A)$.)

It is the condition number $\kappa(A)$ that indicates the amplifying ability of A on the relative error!

Note. We see from discussion above that small condition number is preferred than large ones. Thus, matrices are "well-conditioned" with small κ while "ill-conditioned" otherwise. The process of reducing κ is often called conditioning. For example, pivoting.

• Example. (Recall)

$$\left(\begin{array}{cc|c} \varepsilon & 1 & 1+\varepsilon \\ 1 & 1 & 2 \end{array}\right) \Rightarrow$$

$$\text{Exact: } \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \qquad \text{computed (with round off error): } \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

why? Guassian elimination increases the condition number. Then, small round-off error in the RHS can lead to large error in the solution

$$\frac{\|\vec{e}\|}{\|\vec{x}\|} \leq \kappa(A)\frac{\|\vec{r}\|}{\|\vec{b}\|}$$

Also, small round-off error in the coefficient matrix can lead to large error in the solution.

$$A\vec{x} = \vec{b}, \quad \tilde{A}\tilde{x} = \vec{b}, \quad \Rightarrow$$

$$\frac{\|\vec{x} - \tilde{x}\|}{\|\tilde{x}\|} \leq \kappa(A) \frac{\|A - \tilde{A}\|}{\|A\|}$$

that is, relative error $\frac{\|\vec{e}\|}{\|\tilde{x}\|}$ can be as large as $\kappa(A)\times$ relative error in the matrix $\frac{\|A - \tilde{A}\|}{\|A\|}$

Back to the example. $||A||_\infty = 2,\ ||A^{-1}||_\infty = \frac{2}{1-\varepsilon}$ so $\kappa(A) = \frac{4}{1-\varepsilon} \approx 4$.

However, after one step of Gaussian Elimination w/o pivoting

$$A \to A_1 = \begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{pmatrix}$$

so $||A_1||_\infty = \frac{1}{\varepsilon} - 1,\ ||A_1^{-1}||_\infty = \frac{1}{\varepsilon(1-\varepsilon)}$ and $\kappa(A_1) \approx \frac{1}{\varepsilon^2}$. Much larger than $\kappa(A)$!

Hence, a small change (e.g. round-off error) in the coefficient matrix A_1 or the RHS of the reduced system can lead to a large change in the computed solution – as large as $\kappa(A_1)$ times in terms of relative errors.

Remedy: pivoting.

$$A \to (pivoting) \to \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 1 \\ 0 & 1 - \varepsilon \end{pmatrix} =: A_2$$

with $\kappa(A_2) \approx 4$ almost the same as $\kappa(A)$!

Note. The propagation of errors is related to the stability of a nunerical method. In the case of linear system, the condition number is a key factor in designing stable algorithms.