$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}_{m\times n}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1}}_{b}$$

CX1104: **Linear Algebra for Computing**

Chap. No : **8.4.3**

Lecture : **Eigen and Singular Values**

Topic : **SVD & Pseudoinverse**

Concept : **SVD, Rank and Condition Number**

Instructor: **A/P Chng Eng Siong**

TAs: **Zhang Su, Vishal Choudhari**

Rev: 24th July 2020

SVD and Rank of a Matrix

Given a matrix $A \in R^{m \times n}$:

- No. of linearly independent rows = Row rank
- No. of linearly independent columns = Column rank
- Row rank = Column rank = Rank of matrix A
- Rank(A) = No. of non-zero singular values of A.
- Hence, rank(A) = rank(A^T).

- Matrix A is said to have full rank if its rank equals the largest possible for a matrix of the same dimensions, which is the lesser of the number of rows and columns. For a full rank matrix, rank(A) = min(m, n).
- Matrix A is said to be rank-deficient if it does not have full rank. For a rank-deficient matrix, rank(A) < min(m, n).

Ref: https://en.wikipedia.org/wiki/Rank_(linear_algebra)

MATLAB Command:

rank(A) % applicable to any $m \times n$ matrix

```
A =

     1     2     3
     1     2     3
     1     2     3
     1     2     3

>> rank(A)

ans =

     1
```

Example: Rank Number [be wary!]

```
>> A(1,1) = 1+1e-4

A =

    1.0001    2.0000    3.0000
    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000

>> rank(A)

ans =

     2
```

A very small offset from 1

Rank is computed by number of non-zero singular value, and can be fooled by computation error.

```
>> A(1,1) = 1+1e-12

A =

    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000

>> rank(A)

ans =

     2
```

An even smaller offset from 1

```
>> rank(A,1e-3)

ans =

     1
```

Set a tolerance as second parameter.

```
rank(A,TOL) is the number of singular values of A
that are larger than TOL. By default, TOL = max(size(A)) * eps(norm(A)).
```

SVD and Condition Number of a Matrix

Condition Number

The ratio C of the largest to smallest singular value in the singular value decomposition of a matrix. The base-b logarithm of C is an estimate of how many base-b digits are lost in solving a linear system with that matrix. In other words, it estimates worst-case loss of precision. A system is said to be singular if the condition number is infinite, and ill-conditioned if it is too large, where "too large" means roughly $\log(C) \gtrsim$ the precision of matrix entries.

A condition number for a matrix and computational task measures how sensitive the answer is to perturbations in the input data and to roundoff errors made during the solution process.

When we simply say a matrix is "ill-conditioned", we are usually just thinking of the sensitivity of its inverse and not of all the other condition numbers.

It basically measures how sensitive the matrix is when calculating its inverse.

A problem with a low condition number is said to be **well-conditioned**, while a problem with a high condition number is said to be **ill-conditioned**. In non-mathematical terms, an ill-conditioned problem is one where, for a small change in the inputs (the independent variables or the right-hand-side of an equation) there is a large change in the answer or dependent variable. This means that the correct solution/answer to the equation becomes hard to find. The condition number is a property of the problem.

Ref: https://blogs.mathworks.com/cleve/2017/07/17/what-is-the-condition-number-of-a-matrix/
https://en.wikipedia.org/wiki/Condition_number

MATLAB Command:

cond(A) % applicable to any $m \times n$ matrix

```
A =

     1     2     3
     1     2     3
     1     2     3
     1     2     3

>> rank(A)

ans =

     1

>> cond(A)

ans =

   1.7884e+33
```

```
>> [U,W,V] = svd(A)

U =

   -0.5000   -0.8660    0.0000    0.0000
   -0.5000    0.2887    0.8165   -0.0000
   -0.5000    0.2887   -0.4082   -0.7071
   -0.5000    0.2887   -0.4082    0.7071

W =

    7.4833         0         0
         0    0.0000         0
         0         0    0.0000
         0         0         0

V =

   -0.2673   -0.9636         0
   -0.5345    0.1482   -0.8321
   -0.8018    0.2224    0.5547
```

```
>> num2str(W, 4)

ans =

  4×27 char array

    '7.483            0            0'
    '    0    9.265e-17            0'
    '    0            0    4.184e-33'
    '    0            0            0'
```

NOTE: In the example above, A is a singular matrix, as $det(A) = 0$.

Example: Condition Number [Somewhat Ill-conditioned!]

```
A = [1 2; 1 2; 1 2.001]
[U,E,V] = svd(A)
cond(A)
E(1,1)/E(2,2)
```

```
A =

    1.0000    2.0000
    1.0000    2.0000
    1.0000    2.0010


U =

   -0.5773   -0.4084   -0.7071
   -0.5773   -0.4084    0.7071
   -0.5775    0.8164   -0.0000

E =

    3.8735         0
         0    0.0004
         0         0


V =

   -0.4472   -0.8945
   -0.8945    0.4472

ans =

   1.0609e+04

ans =

   1.0609e+04
```

Example: Condition Number [Very Ill-conditioned!]

```
A = [1 2; 1 2; 1 2]
[U,E,V] = svd(A)
cond(A)
E(1,1)/E(2,2)
```

```
A =

     1     2
     1     2
     1     2


U =

   -0.5774    0.8165   -0.0000
   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071


E =

    3.8730         0
         0    0.0000
         0         0


V =

   -0.4472    0.8944
   -0.8944   -0.4472


ans =

   1.3821e+17


ans =

   1.3821e+17
```

Although theoretically we know that E(2, 2) = 0 exactly, due to numeric representational inaccuracy, E(2, 2) does not exactly equal zero internally on the PC!

```
>> E(2,2)

ans =

   2.8022e-17
```