Lecture 7

Gaussian Elimination with Pivoting

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

July 8, 2014

Naive Gaussian Elimination Algorithm

- Forward Elimination
- + Backward substitution
- = Naive Gaussian Elimination

Goals for today. . .

- Identify *why* our basic GE method is “naive”: identify where the errors come from?
  - division by zero, near-zero
- Propose strategies to eliminate the errors
  - partial pivoting, complete pivoting, scaled partial pivoting
- Investigate the cost: does pivoting cost too much?
- Try to answer “How *accurately* can we solve a system with or without pivoting?”
  - Analysis tools: norms, condition number, . . .

Why is our basic GE “naive”?

Example

$$A = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1e-10 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The Need for Pivoting

Solve:

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} \qquad b = \begin{bmatrix} -4 \\ 5 \\ 7 \\ 7 \end{bmatrix}$$

Note that there is nothing ”wrong” with this system. A is full rank. The solution exists and is unique.
Form the augmented system.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 1 & 2 & 4 & -3 & 5 \\ -3 & -3 & 8 & -2 & 7 \\ -1 & 1 & 6 & -3 & 7 \end{array}\right]$$

The Need for Pivoting

Subtract $1/2$ times the first row from the second row,
add $3/2$ times the first row to the third row,
add $1/2$ times the first row to the fourth row.
The result of these operations is:

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 0 & 5 & -2 & 7 \\ 0 & 3 & 5 & -5 & 1 \\ 0 & 3 & 5 & -4 & 5 \end{array}\right]$$

The *next* stage of Gaussian elimination will not work because there is a zero in the *pivot* location, $\tilde{a}_{22}$.

The Need for Pivoting

Swap second and fourth rows of the augmented matrix.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 3 & 5 & -5 & 1 \\ 0 & 0 & 5 & -2 & 7 \end{array}\right]$$

Continue with elimination: subtract (1 times) row 2 from row 3.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 0 & 0 & -1 & -4 \\ 0 & 0 & 5 & -2 & 7 \end{array}\right]$$

The Need for Pivoting

Another zero has appear in the pivot position. Swap row 3 and row 4.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 0 & 5 & -2 & 7 \\ 0 & 0 & 0 & -1 & -4 \end{array}\right]$$

The augmented system is now ready for backward substitution.

another example

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Example

With Naive GE,

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & (1 - \frac{1}{\varepsilon}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \frac{1}{\varepsilon} \end{bmatrix}$$

Solving for x_1 and x_2 using exact arithmetic we get

$$x_2 = \frac{2 - 1/\varepsilon}{1 - 1/\varepsilon} = 1 + \frac{\varepsilon}{\varepsilon - 1} \approx 1 - \varepsilon$$
$$x_1 = \frac{1 - x_2}{\varepsilon} = \frac{-1}{\varepsilon - 1} = 1 - \frac{\varepsilon}{\varepsilon - 1} \approx 1 + \varepsilon$$

However using finite precision floating point arithmetic for $\varepsilon \approx 10^{-20}$, $x_1 \approx 0$, $x_2 \approx 1$. —Why?

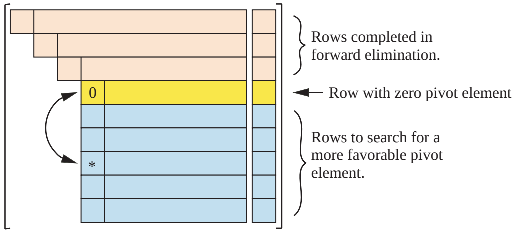
Pivoting Strategies

Partial Pivoting: Exchange only rows

- Exchanging rows does not affect the order of the x_i
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the partial column below the pivot element.
- Partial pivoting is usually sufficient.

Partial Pivoting

To avoid division by zero, swap the row having the zero pivot with one of the rows below it.


Rows completed in forward elimination.
0
Row with zero pivot element
*
Rows to search for a more favorable pivot element.

To minimize the effect of roundoff, always choose the row that puts the largest pivot element on the diagonal, i.e., find i_p such that $|a_{i_p,i}| = \max(|a_{k,i}|)$ for $k = i, \ldots, n$

Partial Pivoting

- Pivoting (that is row exchanges) can be expressed in terms of matrix multiplication
- Do pivoting during elimination, but track row exchanges in order to express pivoting with matrix P
- Let P be all zeros
  - Place a 1 in column j of row 1 to exchange row 1 and row j
  - If no row exchanged needed, place a 1 in column 1 of row 1
  - *Repeat for all rows of* P
- P is a permutation matrix
- Now using pivoting,

$$LU = PA$$

Partial Pivoting Example

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} \qquad b = \begin{bmatrix} -4 \\ 5 \\ 7 \\ 7 \end{bmatrix}$$

Apply the elementary permutation matrix P_1 to permute the first and third rows of A.

$$P_1 * A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} = \begin{bmatrix} -3 & -3 & 8 & -2 \\ 1 & 2 & 4 & -3 \\ 2 & 4 & -2 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix}$$

so we have,

$$P_1 * A * \mathbf{x} = P_1 * \mathbf{b}$$

Partial Pivoting Example

Next apply M_1 an elementary elimination matrix to the previous result,

$$M_1 * P_1 * A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 0 \\ 2/3 & 0 & 1 & 0 \\ -1/3 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -3 & -3 & 8 & -2 \\ 1 & 2 & 4 & -3 \\ 2 & 4 & -2 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} -3 & -3 & 8 & -2 \\ 0 & 1 & 20/3 & -11/3 \\ 0 & 2 & 10/3 & -10/3 \\ 0 & 2 & 10/3 & -7/3 \end{bmatrix}$$

so we have,

$$M_1 * P_1 * A * \mathbf{x} = M_1 * P_1 * \mathbf{b}$$

Partial Pivoting Example

Apply the elementary permutation matrix P_2 to permute the second and third rows.

$$P_2 * M_1 * P_1 * A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -3 & -3 & 8 & -2 \\ 0 & 1 & 20/3 & -11/3 \\ 0 & 2 & 10/3 & -10/3 \\ 0 & 2 & 10/3 & -7/3 \end{bmatrix}$$

$$= \begin{bmatrix} -3 & -3 & 8 & -2 \\ 0 & 2 & 10/3 & -10/3 \\ 0 & 1 & 20/3 & -11/3 \\ 0 & 2 & 10/3 & -7/3 \end{bmatrix}$$

so we have,

$$P_2 * M_1 * P_1 * A * \mathbf{x} = P_2 * M_1 * P_1 * \mathbf{b}$$

Partial Pivoting Example

Next apply M_2 an elementary elimination matrix to the previous result,

$$M_2 * P_2 * M_1 * P_1 * A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1/2 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -3 & -3 & 8 & -2 \\ 0 & 2 & 10/3 & -10/3 \\ 0 & 1 & 20/3 & -11/3 \\ 0 & 2 & 10/3 & -7/3 \end{bmatrix}$$

$$= \begin{bmatrix} -3 & -3 & 8 & -2 \\ 0 & 2 & 10/3 & -10/3 \\ 0 & 0 & 5 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = U$$

so we have,

$$M_2 * P_2 * M_1 * P_1 * A * \mathbf{x} = M_2 * P_2 * M_1 * P_1 * \mathbf{b}$$

and in this example there is no need to apply P_3 then M_3.

Partial Pivoting Example

Note that we can formally write,

$$(M_2 * P_2 * M_1 * P_1) * A = U$$

or

$$A = \left(P_1^{-1} * M_1^{-1} * P_2^{-1} * M_2^{-1}\right) * U = L' * U$$

however the matrix $L' = P_1^{-1} * M_1^{-1} * P_2^{-1} * M_2^{-1}$ is

$$L' = P_1^{-1} * M_1^{-1} * P_2^{-1} * M_2^{-1} = \begin{bmatrix} -2/3 & 1 & 0 & 0 \\ -1/3 & 1/2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 1 \end{bmatrix}$$

and this is NOT lower triangular.

Partial Pivoting Example

But we do note that this matrix L' is a permutation of a lower triangular matrix and specifically that for

$$P = P_2 * P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

then

$$\begin{aligned} P * A &= (P * L') * U \\ &= \left(\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -2/3 & 1 & 0 & 0 \\ -1/3 & 1/2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 1 \end{bmatrix} \right) * U \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2/3 & 1 & 0 & 0 \\ -1/3 & 1/2 & 1 & 0 \\ 1/3 & 1 & 0 & 1 \end{bmatrix} * U \end{aligned}$$

Theorem LUP decomposition

Any non-singular matrix A may be decomposed as

$$L * U = P * A$$

where P is a permutation matrix, L is unit lower triangular, and U is upper triangular.

Python PLU

Consider the Python solution to the problem just considered.We get

$$Ax = \mathbf{b}$$

$$PLU = A$$

```
>>> import numpy as np
>>> from scipy import linalg
>>> A = np.array
    ([[2.,4.,-2.,-2.],[1.,2.,4.,-3.],[-3.,-3.,8.,2.],[-1.,1.,6.,-3.]])

>>> b = np.array([[-4.],[5.],[7.],[7.]])
>>> P,L,U = scipy.linalg.lu(A)
>>> P
array([[ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 1.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  1.]])
>>> L
array([[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00],
       [ -6.66666667e-01,   1.00000000e+00,   0.00000000e+00,
          0.00000000e+00],
       [ -3.33333333e-01,   5.00000000e-01,   1.00000000e+00,
          0.00000000e+00],
       [  3.33333333e-01,   1.00000000e+00,   8.88178420e-17,
          1.00000000e+00]])
>>> U
array([[-3.        , -3.        ,  8.        ,  2.        ],
       [ 0.        ,  2.        ,  3.33333333, -0.66666667],
       [ 0.        ,  0.        ,  5.        , -2.        ],
       [ 0.        ,  0.        ,  0.        , -3.        ]])
>>> np.dot(P, np.dot(L,U))
array([[ 2.,  4., -2., -2.],
       [ 1.,  2.,  4., -3.],
       [-3., -3.,  8.,  2.],
       [-1.,  1.,  6., -3.]])
```

Partial Pivoting: Usually sufficient, but not always

- Partial pivoting is usually sufficient
- Consider

$$\left[\begin{array}{cc|c} 2 & 2c & 2c \\ 1 & 1 & 2 \end{array}\right]$$

With Partial Pivoting, the first row is the pivot row:

$$\left[\begin{array}{cc|c} 2 & 2c & 2c \\ 0 & 1-c & 2-c \end{array}\right]$$

and for large c:

$$\left[\begin{array}{cc|c} 2 & 2c & 2c \\ 0 & -c & -c \end{array}\right]$$

so that $x = 0$ and $y = 1$. (exact is $x = y = 1$ for finite precision floating point arithmetic)

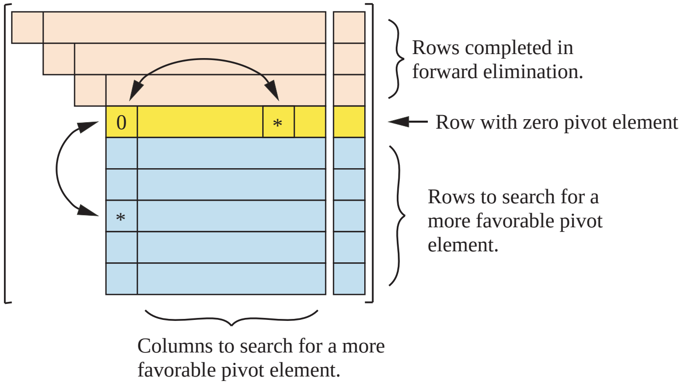
- The pivot is selected as the largest in the column, but it should be the largest relative to the full submatrix.

More Pivoting Strategies

Full (or Complete) Pivoting: Exchange *both* rows and columns

- Column exchange requires changing the order of the x_i
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the pivot row, *and* in all rows below the pivot row, starting the pivot column.
- Full pivoting is less susceptible to roundoff, but the increase in stability comes at a cost of more complex programming (not a problem if you use a library routine) and an increase in work associated with searching and data movement.

Full Pivoting


0
*
*
Rows completed in forward elimination.
Row with zero pivot element
Rows to search for a more favorable pivot element.
Columns to search for a more favorable pivot element.

Full Pivoting

For the matrix (assuming $|c| >> 1$),

$$\left[\begin{array}{cc|c} 2 & 2c & 2c \\ 1 & 1 & 2 \end{array}\right]$$

swap the first and second columns to obtain,

$$\left[\begin{array}{cc|c} 2c & 2 & 2c \\ 1 & 1 & 2 \end{array}\right]$$

and now apply GE,

$$\left[\begin{array}{cc|c} 2c & 2 & 2c \\ 0 & 1 - \frac{1}{c} & 1 \end{array}\right]$$

and for large c:

$$\left[\begin{array}{cc|c} 2c & 2 & 2c \\ 0 & 1 & 1 \end{array}\right]$$

so that $x = 1$ and $y = 1$. (exact is $x = y = 1$)

Theorem LU Decomposition Using Full Pivoting

Any non-singular matrix A may be decomposed as

$$L * U = P * A * Q$$

where P and Q are permutation matrices, L is unit lower triangular, and U is upper triangular.

Application of LU Decomposition — Matrix Inverse

Given a nonsingular matrix A we can find A^{-1} by solving,

$$A * \mathbf{x} = \mathbf{e_i} \quad i = 1, \ldots, n$$

where the identity matrix $I_{n\times n}$ is written in column form as,

$$I = [\mathbf{e_1}|\mathbf{e_2}|\ldots|\mathbf{e_n}]$$

This we are solving,

$$\begin{aligned} A\mathbf{x_1} &= \mathbf{e_1} \\ A\mathbf{x_2} &= \mathbf{e_2} \\ &\vdots \\ A\mathbf{x_n} &= \mathbf{e_n} \end{aligned}$$

and the inverse matrix is therefore (in column format),

Application of LU Decomposition — Matrix Inverse

$$A^{-1} = [\mathbf{x_1}|\mathbf{x_2}|\dots\mathbf{x_n}]$$

Since LU factorization costs $O(\frac{2}{3}n^3)$ (assuming no pivoting) and for the n equations back-solving/forward-solving costs $O(\frac{4}{3}n^3)$ (see pp. 224-225 for details) thus the total cost is $O(2n^3)$.

Application of LU Decomposition — Determinant

Given a matrix A we can perform a factorization,

$$\begin{aligned} P*A &= L*U \text{ thus} \\ det(P)*det(A) &= det(L)*det(U) \\ (-1)^s*det(A) &= 1*\prod_{i=1}^{n} U_{i,i} \end{aligned}$$

where P is a permutation matrix and thus

$$det(P) = \begin{cases} -1 & \text{if the number of elementary permutations is odd} \\ +1 & \text{if the number of elementary permutations is even} \end{cases}$$

L is unit lower triangular and thus,

$$det(L) = 1$$

and U is upper triangular so

$$det(U) = \text{ the product of the diagonal elements of } U$$

Application of LU Decomposition — Determinant

Given a matrix $A = P * L * U$,

```
>>> A = np.array
    ([[2.,4.,-2.,-2.],[1.,2.,4.,-3.],[-3.,-3.,8.,2.],[-1.,1.,6.,-3]])

>>> A
array([[ 2.,  4., -2., -2.],
       [ 1.,  2.,  4., -3.],
       [-3., -3.,  8.,  2.],
       [-1.,  1.,  6., -3.]])
>>> P,L,U = linalg.lu(A)
>>> P
array([[ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 1.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  1.]])
>>> L
array([[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00],
       [ -6.66666667e-01,   1.00000000e+00,   0.00000000e+00,
          0.00000000e+00],
       [ -3.33333333e-01,   5.00000000e-01,   1.00000000e+00,
          0.00000000e+00],
       [  3.33333333e-01,   1.00000000e+00,   8.88178420e-17,
          1.00000000e+00]])
>>> U
array([[-3.        , -3.        ,  8.        ,  2.        ],
       [ 0.        ,  2.        ,  3.33333333, -0.66666667],
       [ 0.        ,  0.        ,  5.        , -2.        ],
       [ 0.        ,  0.        ,  0.        , -3.        ]])
>>> linalg.det(A)
90.0
```

where P is a permutation matrix and thus

$$det(P) = 1(\text{ product of even number of elementary permutations})$$

L is unit lower triangular and thus,

$$det(L) = 1$$

and U is upper triangular so

$$det(U) = -3.0 * 2.0 * 5.0 * -3.0 = 90.0$$

Geometric Interpretation of Singularity

Consider a 2×2 system describing two lines that intersect

$$\begin{aligned} y &= -2x + 6 \\ y &= \frac{1}{2}x + 1 \end{aligned}$$

The matrix form of this equation is

$$\begin{bmatrix} 2 & 1 \\ -1/2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 1 \end{bmatrix}$$

The equations for two **parallel** but **not intersecting** lines are

$$\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

Here the coefficient matrix is singular ($\mathrm{rank}(A) = 1$), and the system is inconsistent

Geometric Interpretation of Singularity

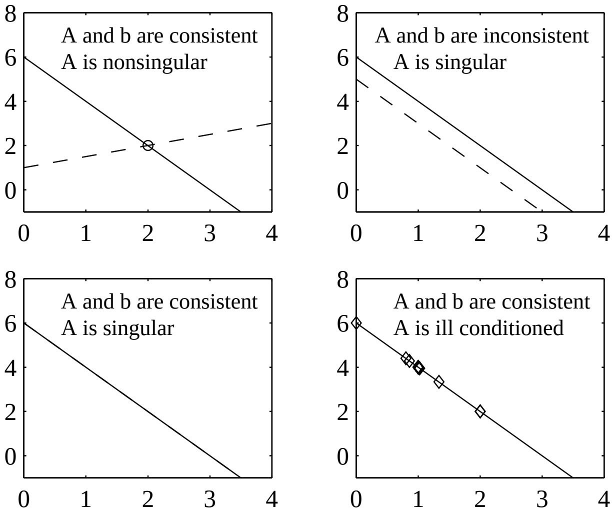
The equations for two **parallel** and **coincident** lines are

$$\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

The equations for two **nearly parallel** lines are

$$\begin{bmatrix} 2 & 1 \\ 2+\delta & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6+\delta \end{bmatrix}$$

Geometric Interpretation of Singularity



Effect of Perturbations to b

Consider the solution of a 2×2 system where

$$b = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

One expects that the *exact* solutions to

$$Ax = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix} \qquad \text{and} \qquad Ax = \begin{bmatrix} 1 \\ 0.6667 \end{bmatrix}$$

will be different. Should these solutions be a **lot different** or a **little different**?

Norms

Vectors:

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n| = \sum_{i=1}^{n} |x_i|$$

$$\|x\|_\infty = \max\left(|x_1|, |x_2|, \ldots, |x_n|\right) = \max_i\left(|x_i|\right)$$

Matrices:

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|u\|=1} \|Au\|$$

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

$$\|A\|_1 = \max_{1\leqslant j\leqslant n} \sum_{i=1}^{m} |a_{ij}|$$

$$\|A\|_\infty = \max_{1\leqslant i\leqslant m} \sum_{j=1}^{n} |a_{ij}|$$

Properties of Vector Norms

Every vector norm must satisfy the following properties.

$$\begin{aligned} \|x\| &> 0 \text{ for } x \neq \mathbf{0} \text{ (where } \mathbf{0} \text{ is the zero vector)} \\ \|\alpha x\| &= |\alpha|\|x\| \text{ for any scalar } \alpha \in \mathbb{R} \\ \|x+y\| &\leqslant \|x\| + \|y\| \text{ triangle inequality} \end{aligned}$$

The vector norm $\|x\|_p$ for $p \geqslant 1$, defined on the previous slide, satisfy the following properties.

$$\|x\|_q \leqslant \|x\|_p \text{ for } 1 \leqslant p \leqslant q \leqslant +\infty$$

and

$$\begin{aligned} \|x\|_1 &\leqslant \sqrt{n}\|x\|_2 \\ \|x\|_2 &\leqslant \sqrt{n}\|x\|_\infty \\ \|x\|_1 &\leqslant n\|x\|_\infty \end{aligned}$$

Properties of Matrix Norms

Every matrix norm must satisfy the following properties.

$$\|A\| > 0 \text{ for } A \neq \mathbf{0} \text{ (where } \mathbf{0} \text{ is the zero matrix)}$$
$$\|\alpha A\| = |\alpha| \|A\| \text{ for any scalar } \alpha \in \mathbb{R}$$
$$\|A + B\| \leqslant \|A\| + \|B\| \text{ triangle inequality}$$

A Matrix norm created from a vector norm by the formula,

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

is called an **induced** matrix norm. Induced matrix norms satisfy the following properties.

$$\|Ax\| \leqslant \|A\| \|x\|$$
$$\|AB\| \leqslant \|A\| \|B\|$$
$$\|I\| = 1 \text{ where } I \text{ is the identity matrix}$$

Effect of Perturbations to b

Perturb b with δb such that

$$\frac{\|(b+\delta b)-b\|}{\|b\|} = \frac{\|\delta b\|}{\|b\|} \ll 1,$$

The perturbed system is

$$A(x+\delta x_b) = b + \delta b$$

Analysis shows (see next two slides for proof) that

$$\frac{\|\delta x_b\|}{\|x\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|} \tag{1}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_b\|}{\|x\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Effect of Perturbations to b (Proof)

Let $x + \delta x_b$ be the *exact* solution to the perturbed system

$$A(x + \delta x_b) = b + \delta b \tag{2}$$

Expand

$$Ax + A\delta x_b = b + \delta b$$

Subtract Ax from left side and b from right side since $Ax = b$

$$A\delta x_b = \delta b$$

Left multiply by A^{-1}

$$\delta x_b = A^{-1}\delta b \tag{3}$$

Effect of Perturbations to b (Proof, p. 2)

Take norm of equation (3)

$$\|\delta x_b\| = \|A^{-1}\,\delta b\|$$

Applying consistency requirement of matrix norms

$$\|\delta x_b\| \leqslant \|A^{-1}\|\|\delta b\| \tag{4}$$

Similarly, $Ax = b$ gives $\|b\| = \|Ax\|$, and

$$\|b\| \leqslant \|A\|\|x\| \tag{5}$$

Rearrangement of equation (5) yields

$$\frac{1}{\|x\|} \leqslant \frac{\|A\|}{\|b\|} \tag{6}$$

Effect of Perturbations to b (Proof)

Multiply Equation (5) by Equation (4) to get

$$\frac{\|\delta x_b\|}{\|x\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|} \tag{7}$$

Summary:

If $x + \delta x_b$ is the *exact* solution to the perturbed system

$$A(x + \delta x_b) = b + \delta b$$

then

$$\frac{\|\delta x_b\|}{\|x\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Condition Number of Matrix

Definition

The condition number of a matrix A is defined by $\kappa(A) = ||A|| * ||A^{-1}||$

Remember our definition of condition number of a problem from lecture 3, how is this related?

Relative Condition Number in higher dimensions

Given a function $G : \mathbb{R}^n \to \mathbb{R}^n$,suppose we wish to compute $y = G(x)$. How sensitive is the solution to changes in x? We can measure this sensitivity by:

- Relative Condition Number = $\lim_{||h|| \to 0} \frac{\frac{||G(x+h) - G(x)||}{||G(x)||}}{\frac{||h||}{||x||}}$

The Taylor Series for G is,

$$G(x+h) = G(x) + G'(x) * h + ...$$

where $G'(x)$ is the Jacobian. Thus, we get,

Condition Number of Matrix

$$\text{Relative Condition Number} = \lim_{||h||\to 0} \frac{||G(x+h) - G(x)||\; ||x||}{||h||\; ||G(x)||} = \frac{||G'(x)||\; ||x||}{||G(x)||}$$

Our problem is to compute $x = A^{-1} * b$ (G is A^{-1}, x in $G(x)$ is b and the y in $y = G(x)$ is x in this analysis). Since the Jacobian of $G(x) = A^{-1}b$ is A^{-1} we can substitute into the above formula to get,

$$\text{Relative Condition Number} = \frac{||A^{-1}||\; ||b||}{||x||}$$

But we have

$$||Ax|| \leqslant ||A||\; ||x||$$

or

$$\frac{||b||}{||x||} \leqslant ||A||$$

and after substitution we have,

$$\text{Relative Condition Number} = \frac{||A^{-1}||\; ||b||}{||x||} \leqslant ||A^{-1}||\; ||A|| \; = \kappa(A)$$

Effect of Perturbations to A

Perturb A with δA such that

$$\frac{\|(A+\delta A)-A\|}{\|A\|} = \frac{\|\delta A\|}{\|A\|} \ll 1,$$

The perturbed system is

$$(A+\delta A)(x+\delta x_A) = b$$

Analysis shows that

$$\frac{\|\delta x_A\|}{\|x+\delta x_A\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_A\|}{\|x+\delta x_A\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Effect of Perturbations to both A and b

Perturb both A with δA and b with δb such that

$$\frac{\|\delta A\|}{\|A\|} \ll 1 \quad \text{and} \quad \frac{\|\delta b\|}{\|b\|} \ll 1$$

The perturbation satisfies

$$(A + \delta A)(x + \delta x) = b + \delta b$$

Analysis shows that

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A\| \|A^{-1}\|}{1 - \|A\| \|A^{-1}\| \frac{\|\delta A\|}{\|A\|}} \left[\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right]$$

Thus, the effect of the perturbation is small *only if* $\|A\| \|A^{-1}\|$ is small.

$$\frac{\|\delta x\|}{\|x\|} \ll 1 \quad \text{only if} \quad \|A\| \|A^{-1}\| \sim 1$$

Condition number of A

The **condition number**

$$\kappa(A) \equiv \|A\|\|A^{-1}\|$$

indicates the sensitivity of the solution to perturbations in A and b. The condition number can be measured with any p-norm.
The condition number is always in the range

$$1 \leqslant \kappa(A) \leqslant \infty$$

- $\kappa(A)$ *is a mathematical property of* A
- *Any algorithm will produce a solution that is sensitive to perturbations in* A *and* b *if* $\kappa(A)$ *is large.*
- *In exact math a matrix is either singular or non-singular.* $\kappa(A) = \infty$ *for a singular matrix*
- $\kappa(A)$ *indicates how close* A *is to being numerically singular.*
- *A matrix with large* κ *is said to be* ***ill-conditioned***

Condition number of A Geometric view

Note that if we denote $y = A^{-1}x$ so that $Ay = x$ we can write,

$$\begin{aligned} \|A^{-1}\| &= \max_{x\neq 0} \frac{\|A^{-1}x\|}{\|x\|} \\ &= \max_{y\neq 0} \frac{\|y\|}{\|Ay\|} \\ &= \max_{y\neq 0} \frac{1}{\frac{\|Ay\|}{\|y\|}} \\ &= \frac{1}{\min_{y\neq 0} \frac{\|Ay\|}{\|y\|}} \end{aligned}$$

so

$$\kappa(A) = \frac{\max_{x\neq 0} \frac{\|Ax\|}{\|x\|}}{\min_{y\neq 0} \frac{\|Ay\|}{\|y\|}}$$

and thus the condition number κ measures the ratio of the largest stretching to smallest stretching of any non-zero vector by the matrix.

The Residual

Let $\hat{x}$ be the numerical solution to $Ax = b$. $\hat{x} \neq x$ (x is the exact solution) because of roundoff.
The **residual** measures how close $\hat{x}$ is to satisfying the original equation

$$r = b - A\hat{x}$$

It is not hard to show from equation (1) that

$$\frac{\|\hat{x} - x\|}{\|x\|} \leqslant \kappa(A) \frac{\|r\|}{\|b\|}$$

Small $\|r\|$ does not guarantee a small $\|\hat{x} - x\|$.
If $\kappa(A)$ is large the $\hat{x}$ returned by Gaussian elimination and back substitution (or any other solution method) is *not* guaranteed to be anywhere near the true solution to $Ax = b$.

Computational Stability

In Practice, applying Gaussian elimination with partial pivoting and back substitution to $Ax = b$ gives the **exact solution**, $\hat{x}$, to the **nearby problem**(see INC p. 239)

$$(A + \delta A)\hat{x} = b \quad \text{where} \quad \|\delta A\|_\infty \lesssim \frac{\varepsilon_m}{2}\|A\|_\infty$$

Gaussian elimination with partial pivoting and back substitution "gives exactly the right answer to nearly the right question."
— Trefethen and Bau

Computational Stability

An algorithm that gives the exact answer to a problem that is near to the original problem is said to be **backward stable**. Algorithms that are not backward stable will tend to amplify roundoff errors present in the original data. As a result, the solution produced by an algorithm that is not backward stable will not necessarily be the solution to a problem that is close to the original problem.

Gaussian elimination **without** partial pivoting is *not* backward stable for arbitrary A.

If A is symmetric and positive definite, then Gaussian elimination without pivoting in backward stable.

For a fixed vector b and for different matrices A the exact solution of $Ax = b$ can be written as $x = A^{-1}b$. With finite precision arithmetic the solution is $\hat{x}$. The matrix $A + \delta A$ produces the solution of $(A + \delta A)\hat{x} = b$ with exact arithmetic.

$$\begin{tikzcd}
\mathbf{A} \arrow[r, "\text{exact}"] \arrow[dr] & \mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \\
\mathbf{A}+\delta\mathbf{A} \arrow[r, "\text{exact}"] & \hat{\mathbf{x}} = (\mathbf{A}+\delta\mathbf{A})^{-1}\mathbf{b}
\end{tikzcd}$$

Rules of Thumb

From

$$(A+\delta A)\hat{x} = b \quad \text{where} \quad \|\delta A\|_\infty \lesssim \frac{\varepsilon_m}{2}\|A\|_\infty$$

and the definition of the residual,

$$r = b - A\hat{x}$$

we can derive the following approximation,

$$\frac{\|\hat{x}-x\|_\infty}{\|x\|_\infty} \lesssim \frac{\varepsilon_m}{2}\kappa(A)$$

- Applying Gaussian elimination with partial pivoting and back substitution to $Ax = b$ yields a numerical solution $\hat{x}$ such that the residual vector $r = b - A\hat{x}$ is small *even if* the $\kappa(A)$ is large.
- If A and b are stored to machine precision ε_m, the numerical solution to $Ax = b$ by any variant of Gaussian elimination is correct to d digits where

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Rules of Thumb

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Example:
Python computations have $\varepsilon_m \approx 2.2 \times 10^{-16}$. For a system with $\kappa(A) \sim 10^{10}$ the elements of the solution vector will have

$$\begin{aligned} d &= |\log_{10}(2.2 \times 10^{-16})| - \log_{10}\left(10^{10}\right) \\ &\approx 16 - 10 \\ &\approx 6 \end{aligned}$$

correct (decimal) digits

Summary of Limits to Numerical Solution of $Ax = b$

1. $\kappa(A)$ indicates how close A is to being numerically singular
2. If $\kappa(A)$ is "large", A is **ill-conditioned** and *even the best* numerical algorithms will produce a solution, $\hat{x}$ that cannot be guaranteed to be close to the true solution, x
3. In practice, Gaussian elimination with partial pivoting and back substitution produces a solution with a small residual

$$r = b - A\hat{x}$$

even if $\kappa(A)$ is large.

The Backslash Operator

Consider the scalar equation

$$5x = 20 \qquad \Longrightarrow \qquad x = (5)^{-1}20$$

The extension to a system of equations is, of course

$$Ax = b \qquad \Longrightarrow \qquad x = A^{-1}b$$

where $A^{-1}b$ is the formal solution to $Ax = b$
In MATLAB notation the system is solved with

```
    x = A\b
```

The Backslash Operator

Given an $n \times n$ matrix A, and an $n \times 1$ vector b the \ operator performs a sequence of tests on the A matrix. MATLAB attempts to solve the system with the method that gives the least roundoff and the fewest operations.
When A is an $n \times n$ matrix:

1. MATLAB examines A to see if it is a permutation of a triangular system

   If so, the appropriate triangular solve is used.

2. MATLAB examines A to see if it *appears* to be symmetric and positive definite.

   If so, MATLAB attempts a Cholesky factorization
   and two triangular solves.

3. If the Cholesky factorization fails, or if A does not appear to be symmetric,

   MATLAB attempts an LU factorization
   and two triangular solves.