Lecture 8

Gaussian Elimination with Pivoting

L. Olson

Department of Computer Science
University of Illinois at Urbana-Champaign

September 20, 2006

Recall: Solving Diagonal Systems

Listing 1: Diagonal System Solution

```
given A, b
for i = 1...n
    x_i = b_i / a_{i,i}
end
```

In Matlab:

```
>> A = ...                % A is a diagonal matrix
>> b = ...
>> x = b./diag(A)
```

This is the *only* place where element-by-element division (`.*`) has anything to do with solving linear systems of equations.

Recall: Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

Listing 2: backward substitution (page 270)

```
given A (upper △), b
x_n = b_n/a_nn
for i = n − 1...1
    s = b_i
    for j = i + 1...n
        s = s − a_{i,j} x_j
    end
    x_i = s/a_{i,i}
end
```

Recall: Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

Listing 3: backward substitution (page 270)

```
given A (upper △), b
x_n = b_n/a_nn
for i = n − 1...1
    s = b_i
    for j = i + 1...n
        s = s − a_{i,j} x_j
    end
    x_i = s/a_{i,i}
end
```

Using forward or backward substitution is sometimes referred to as performing a **triangular solve**.

Forward Elimination Algorithm

Listing 4: Forward Elimination

```
given A, b

for k = 1...n-1
   for i = k+1...n
      xmult = a_ik / a_kk
      a_ik = xmult
      for j = k+1...n
         a_ij = a_ij - (xmult) a_kj
      end
      b_i = b_i - (xmult) b_k
   end
end
```

Naive Gaussian Elimination Algorithm

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

why is this ”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,

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

Solving for x_1 and x_2 we get

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

$$x_2 = \frac{1 - x_2}{\varepsilon}$$

For $\varepsilon \approx 10^{-20}$, $x_1 \approx 0$, $x_2 \approx 1$

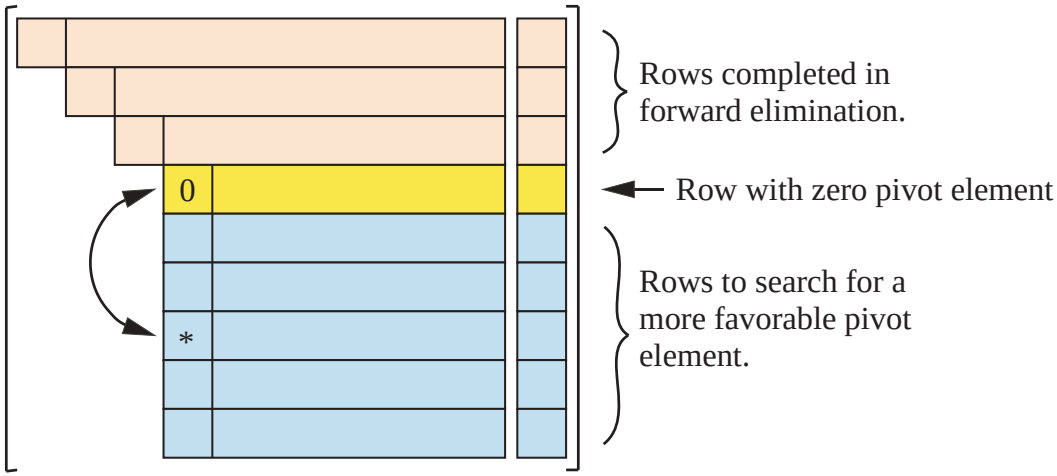
Pivoting Strategies

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.



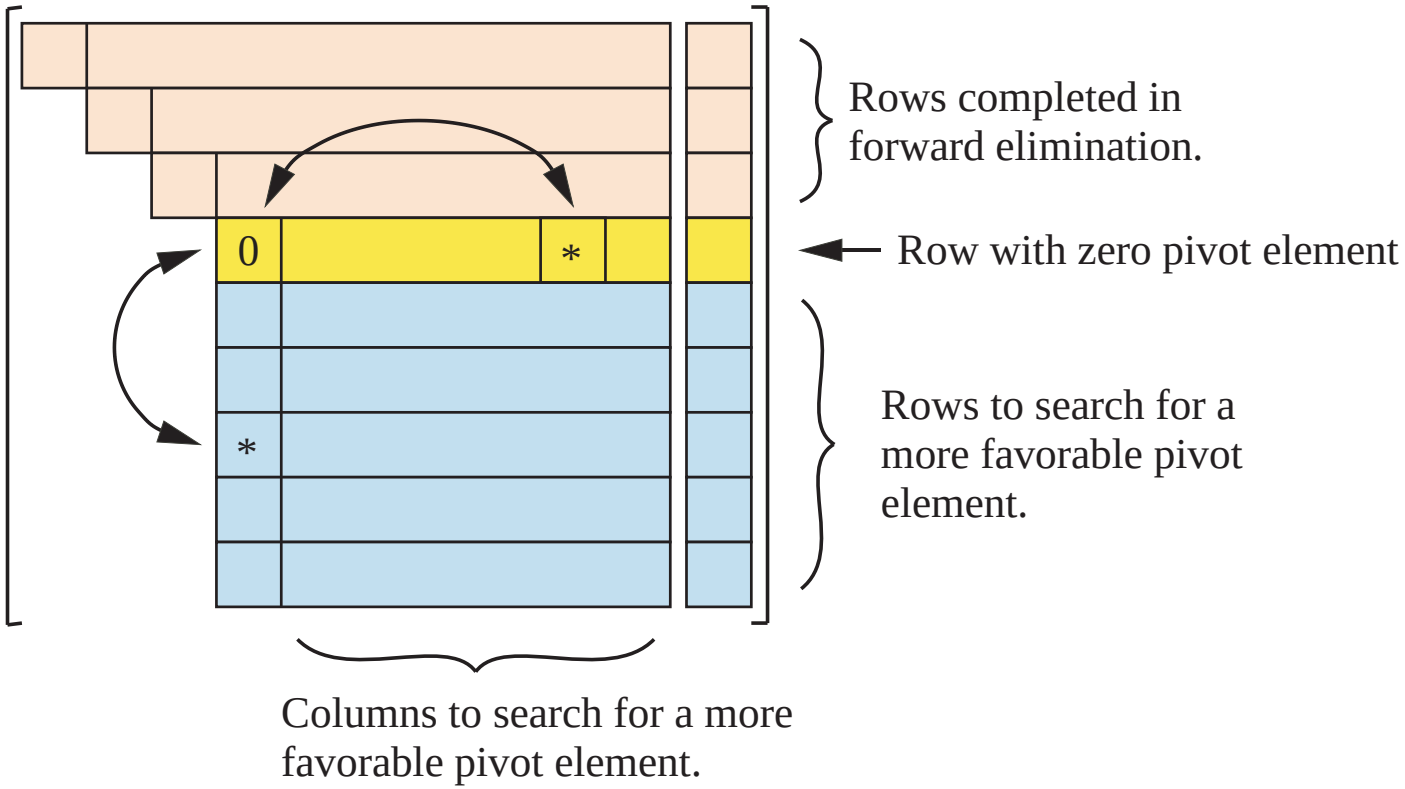
To minimize the effect of roundoff, always choose the row that puts the largest pivot element on the diagonal, i.e., find i_p such that $|a_{i_p,i}| = \max(|a_{k,i}|)$ for $k = i, \ldots, n$

Pivoting Strategies (2)

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



Scaled Partial Pivoting

We simulate full pivoting by using a scale with partial pivoting.

- pick pivot element as the largest **relative** entry in the column (relative to the other entries in the row)
- do not swap, just keep track of the order of the pivot rows
- call this vector $\ell = [\ell_1, \ldots, \ell_n]$.

SPP Process

1. Determine a scale vector **s**. For each row

$$s_i = \max_{1 \leqslant j \leqslant n} |a_{ij}|$$

2. initialize $\ell = [\ell_1, \ldots, \ell_n] = [1, \ldots, n]$.
3. select row j to be the row with the largest ratio

$$\frac{|a_{\ell_i 1}|}{s_{\ell_i}} \qquad 1 \leqslant i \leqslant n$$

4. swap ℓ_j with ℓ_1 in ℓ
5. Now we need $n-1$ multipliers for the first column:

$$m_1 = \frac{a_{\ell_i 1}}{a_{\ell_1 1}}$$

6. So the index to the rows are being swapped, NOT the actual row vectors which would be expensive
7. finally use the multiplier m_1 times row ℓ_1 to subtract from rows ℓ_i for $2 \leqslant i \leqslant n$

SPP Process continued

1. For the second column in forward elimination, we select row j that yields the largest ratio of
$$\frac{|a_{\ell_i,1}|}{s_{\ell_i}} \qquad 2 \leqslant i \leqslant n$$
2. swap ℓ_j with ℓ_2 in ℓ
3. Now we need $n-2$ multipliers for the second column:
$$m_2 = \frac{a_{\ell_i,2}}{a_{\ell_2 2}}$$
4. finally use the multiplier m_2 times row ℓ_2 to subtract from rows ℓ_i for $3 \leqslant i \leqslant n$
5. the process continues for row k
6. note: scale factors are *not* updated

An Example

Consider

$$\begin{bmatrix} 2 & 4 & -2 \\ 1 & 3 & 4 \\ 5 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -1 \\ 2 \end{bmatrix}$$

Back Substitution...

1. The first equation corresponds to the last index ℓ_n:

$$a_{\ell_n n} x_n = b_{\ell_n} \Rightarrow x_n = \frac{b_{\ell_n}}{a_{\ell_n n}}$$

2. The second equations corresponds to the second to last index ℓ_{n-1}:

$$x_{n-1} = \frac{1}{a_{\ell_{n-1} n-1}} \left(b_{\ell_{n-1}} - a_{\ell_{n-1} n} x_n \right)$$

The Algorithms

Listing 5: (forward) GE with SPP

```
1  Initialize ℓ = [1,...,n]
2  Set s to be the max of rows
3  for k = 1 to n
4     rmax = 0
5     for i = k to n
6        r = |a_{ℓ_i k} / s_{ℓ_i}|
7        if(r > rmax)
8           rmax = r
9           j = i
10    end
11    swap ℓ_j and ℓ_k
12    for i = k + 1 to n
13       xmult = a_{ℓ_i k} / a_{ℓ_k k}
14       a_{ℓ_ik} = xmult
15       for j = k + 1 to n
16          a_{ℓ_i j} = a_{ℓ_i j} − xmult · a_{ℓ_k j}
17       end
18    end
19 end
```

See *Gauss* algorithm on page 285 of NMC.

The Algorithms

Note: the multipliers are stored in the location $a_{\ell_i k}$ in the text

Listing 6: (back solve) GE with SPP

```
1   for k = 1 to n − 1
2      for i = k + 1 to n
3         b_{ℓ_i} = b_{ℓ_i} − a_{ℓ_i k} b_{ℓ_k}
4      end
5   end
6   x_n = b_{ℓ_n} / a_{ℓ_n n}
7   for i = n − 1 down to 1
8      sum = b_{ℓ_i}
9      for j = i + 1 to n
10        sum = sum − a_{ℓ_i j} x_j
11     end
12  end
```

See *Gauss* algorithm on page 287 of NMC.

Geometric Interpretation of Singularity

Consider a 2×2 system describing two lines that intersect

$$y = -2x + 6$$
$$y = \frac{1}{2}x + 1$$

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

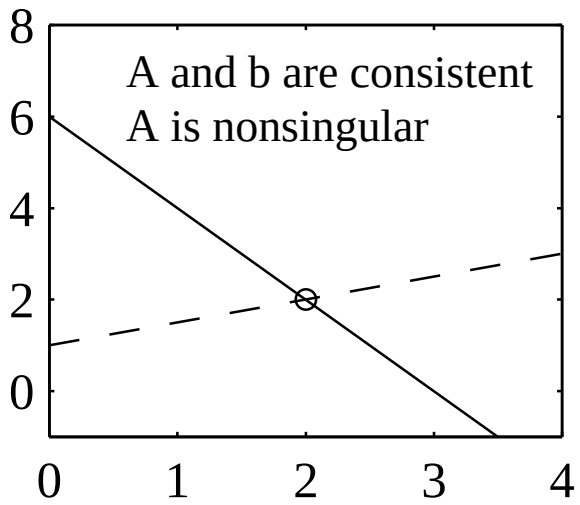
The equations for two **parallel** and **coincident** lines are

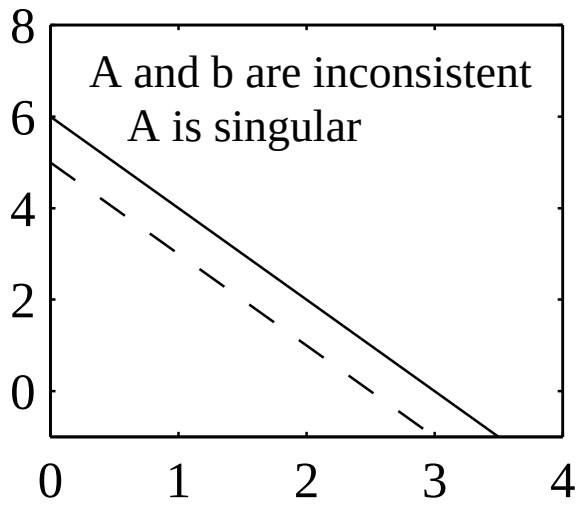
$$\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

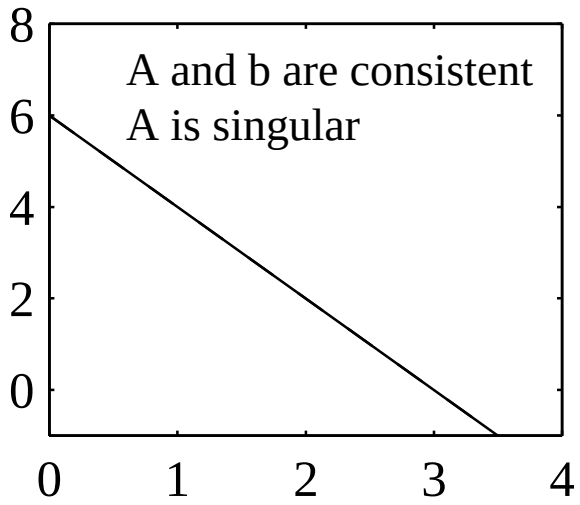
The equations for two **nearly parallel** lines are

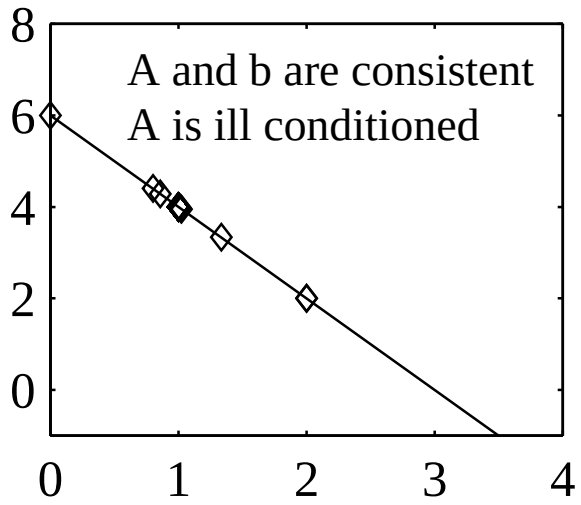
$$\begin{bmatrix} 2 & 1 \\ 2+\delta & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6+\delta \end{bmatrix}$$

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**?

Effect of Perturbations to b

Perturb b with δb such that

$$\frac{\|\delta b\|}{\|b\|} \ll 1,$$

The perturbed system is

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

The perturbations satisfy

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

Analysis shows (see next two slides for proof) that

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

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

$$A(x + \delta x_b) = b + \delta b \tag{1}$$

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}

$$\delta x_b = A^{-1}\delta b \tag{2}$$

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

Take norm of equation (2)

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

Applying consistency requirement of matrix norms

$$\|\delta x\| \leqslant \|A^{-1}\|\|\delta b\| \tag{3}$$

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

$$\|b\| \leqslant \|A\|\|x\| \tag{4}$$

Rearrangement of equation (4) yields

$$\frac{1}{\|x\|} \leqslant \frac{\|A\|}{\|b\|} \tag{5}$$

Effect of Perturbations to b (Proof)

Multiply Equation (4) by Equation (3) to get

$$\frac{\|\delta x_b\|}{\|x\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|} \tag{6}$$

Summary:

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

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

then

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

Effect of Perturbations to A

Perturb A with δA such that

$$\frac{\|\delta A\|}{\|A\|} \ll 1,$$

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

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

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

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

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***

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**

$$(A + E)\hat{x} = b \quad \text{where} \quad \|E\|_\infty \leqslant \varepsilon_m \|A\|_\infty$$

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.

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 that

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

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 $Ax = b$.

Rules of Thumb

- 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:
MATLAB 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

$$\begin{aligned} d &= |\log_{10}(2.2 \times 10^{-16})| - \log_{10}(10^{10}) \\ &= 16 - 11 \\ &= 5 \end{aligned}$$

correct 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

```
1     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.