ME 350 Lecture 8 Solving Systems of Linear Equations
Winter 2017

# 1 Learning Objectives

At the end of this class you will be able to

- Set up a system of $n$ equations in $n$ unknowns:
  create $A$ and $b$ in $Ax = b$.
- State the condition that must hold if a solution *exists*. Explain this condition in terms of the column space of $A$.
- State the condition that must hold if the solution to $Ax = b$ *exist and is unique* when $A$ is $n \times n$
- explain the significance of the condition number
- compute and interpret the residual
- explain a rule of thumb estimate for the number of significant digits in the solution to a linear system of equations.

# 2 Standard Form

The standard form for a system of linear equations is

$$Ax = b$$

$A$ is a matrix, usually square, $x$ is a column vector, $b$ is a column vector. The unknowns are elements of $x$. Multipliers of $x$ (coefficients of $x$) are elements of $A$. Everything else must be known and goes in $b$.

## Putting a system into Standard Form

1. Write equations in "natural" form, i.e., the form that arises from the given problem. Don't try to write $A$, $x$ and $b$ just by looking at your equations.
2. Identify unknowns and order them. Associate this ordered list of unknowns with the elements of $x$.
3. Manipulate equations:
   - Gather all multipliers of each unknown on left hand side
   - Move all leftovers to right hand side
4. Transfer to matrix form.

# 3 Requirements for a solution

- $b$ must lie in range($A$) for solution to exist. In other words, $b$ must lie in the column space of $A$.

  **$Ax = b$ is a statement that $x$ is a set of coefficients that are used to construct $b$ from a linear combination of the columns of $A$.**

- if $A$ is not full rank, solution will not be unique
- For $n \times n$ systems
  - If rank($A$) $= n$, the $n$ columns of $A$ are linearly independent, so columns of $A$ span $\mathbf{R}^n$. Therefore, any $n$ element vector, $b$, must lie in range($A$), and a solution exists.
  - If rank($A$) $= n$ the solution is also unique

# 4 Formal Solution

The formal solution to $Ax = b$ is $x = A^{-1}b$, when $A$ is $n \times n$.

- If $A^{-1}$ does not exist, then $A$ is *singular*
- If $A^{-1}$ does exist, then $A$ is *nonsingular*

**Do not use $x = A^{-1}b$ in MATLAB, e.g., `x = inv(A)*b`. Use the backslash operator instead.**

# 5 Gaussian Elimination

Gaussian Elimination: the goal is to transform a system so that the coefficient matrix is triangular. The triangular system is then solved by backward substitution.

## 5.1 Solving $Ax = b$ in MATLAB

The "backslash" operator, \, is used to solve linear systems of equations in MATLAB. 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.

Consider the linear system defined by

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

The following statements define matrix `A` and column vector `b`

```
>> A = [2 4 -2 2; 1 2 4 -3; -3 -3 8 -2; -1 1 6 -3]
A =
     2     4    -2     2
     1     2     4    -3
    -3    -3     8    -2
    -1     1     6    -3

>> b = [-4; 5; 7; 7]
b =
    -4
     5
     7
     7
```

Remember that row elements are separated by spaces or commas, and columns are separated by semicolons. With `A` and `b` defined as above, the solution to $Ax = b$ is

```
>> x = A\b
x =
   -0.7778
    0.2222
    0.3333
   -1.3333
```

*Do not* use the `inv` command. Although one can usually get away with solving $Ax = b$ with the `inv` commmand, it is a bad habit for the following reasons.

- Solving $Ax = b$ with `inv` takes more operations than `x = A\b`. The difference in operation count becomes significant as $n$ increases. ($A$ is $n \times n$.)
- The backslash operator examines the properties of `A` before it chooses one of several solution methods. The process is represented graphically in Figure 1. As long as `A` is full rank and well-conditioned, the different methods are equivalent, but the solution algorithm chosen by the backslash algorithm will be as efficient as possible and use the least amount of effort possible.
- Solving $Ax = b$ with `inv` is more susceptible to roundoff than `x = A\b`.

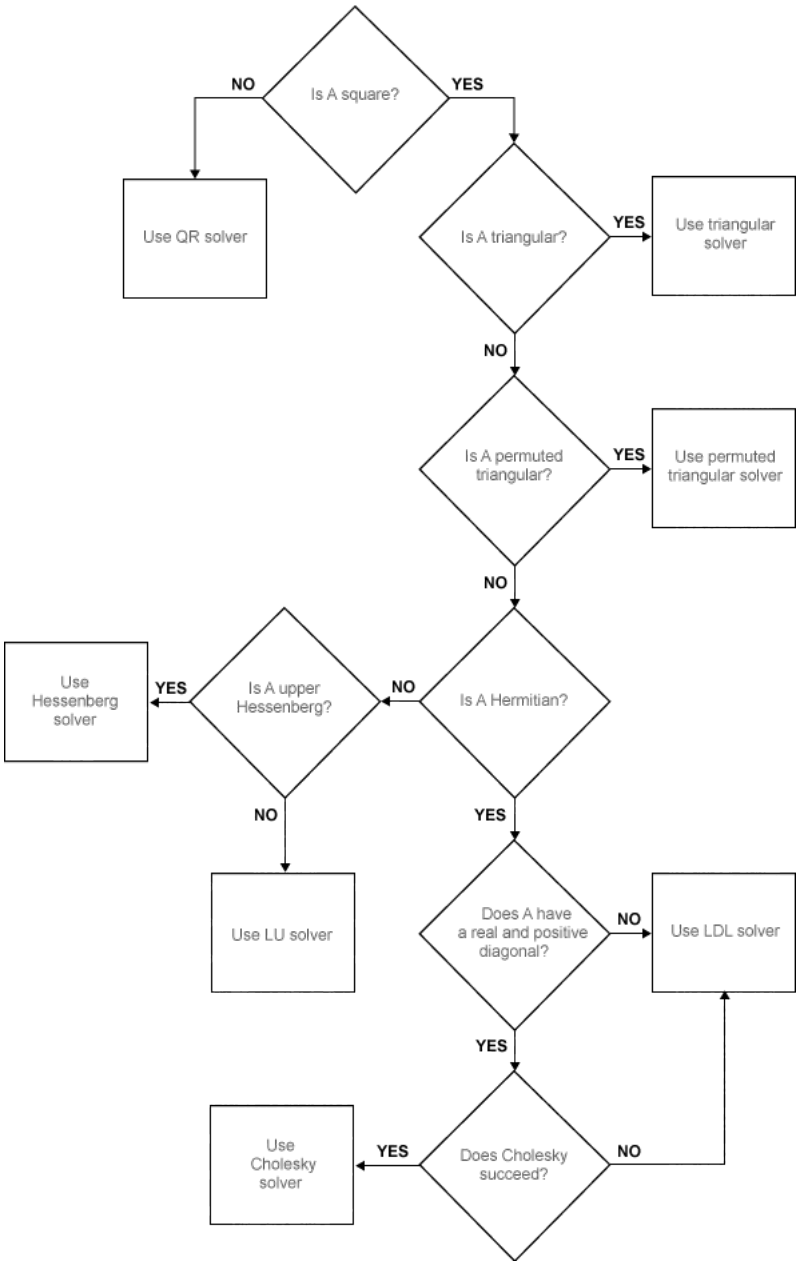


Figure 1: Analysis of the coefficient matrix and subsequent solution methods used by the backslash operator. Image from the Mathworks `https://www.mathworks.com/help/releases/R2016b/matlab/ref/mldivide_full.png`.

# 6 Numerical Accuracy and Stability

Solving a linear system involves $\mathcal{O}(n^3)$ floating point operations when $A$ is $n \times n$. Therefore, as $n$ increases, the work increases rapidly. For large $n$ roundoff errors can also reduce the accuracy of the solution. However, even for modest-sized systems, roundoff can be important if the coefficient matrix is *ill-conditioned*. In this section we introduce the *condition number*, which is a scalar value that indicates how close the coefficient matrix is to being singular.

## 6.1 Condition Number

- In exact arithmetic a square matrix is either singular or not.
- In floating point arithmetic, a matrix can be *more* or *less* singular. The condition number of a matrix is "large" when the matrix is nearly singular
- The numerical solution to $Ax = b$ is $\hat{x}$. Because of roundoff, $\hat{x} \neq x$, i.e., the numerical solution is not exactly equal to the true (perfect, analytical) solution. We hope that $\hat{x}$ is *close to* $x$, i.e. we want a small $\|\hat{x} - x\|$.
- The *condition number* of matrix $A$ is

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

  where $\|A\|$ is the *matrix* norm.

  Note that matrix norms are not obtained by applying the formulas for vector norms to the elements of $A$. In general, matrix norms are expensive to compute. The built-in `condest` and `rcond` functions allow us to estimate the condition number of matrices without incurring the full computational cost of computing the matrix norms.

  In general

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

  and $\kappa(A) = \infty$ for a singular matrix. Thus,

  > **The magnitude of $\kappa(A)$ indicates how close $A$ is to being singular.**

  A large $\kappa(A)$ means that $A$ is *ill conditioned*.
- When $A$ is *ill conditioned*, small changes in the elements of $A$ and $b$ can have a big effect on the numerical solution to $Ax = b$.
- A matrix that has a large $\kappa$ is susceptible to roundoff in a way that makes the solution to $Ax = b$ potentially unreliable regardless of how $x$ is computed. In other words,

  > **The condition number is a characteristic of the problem being solved, not the algorithm used to obtain the solution.**
- In exact arithmetic (infinite number of digits) a matrix is either singular ($A^{-1}$ does not exist) or nonsingular ($A^{-1}$ exists). Thus, the condition number has no bearing on accuracy of solution obtained in exact arithmetic, i.e. in symbolic form.

## 6.2 Stability

"Good" and "bad" algorithms — Backward stability

- A good algorithm will not amplify roundoff errors present in the input data. Roundoff is always present. An unstable algorithm makes the effect of roundoff worse.

  Roundoff in the input occurs when elements of $A$ and or $b$ come from (uncertain) experimental data, or when the elements of $A$ and $b$ are stored with limited precision (roundoff)

- A good algorithm is *backward stable*, i.e. it produces the exact solution to a problem that is close to the original problem.

- Gaussian elimination *without* pivoting is *not* backward stable. The algorithm cannot be guaranteed to give small $\|\hat{x} - x\|$ even if the problem has a small $\kappa(A)$ and even if no zero pivot is encountered

- Gaussian elimination with partial pivoting (GEPP) *is* backward stable. The $\hat{x}$ vector produced by applying GEPP to $Ax = b$ is the exact solution to

$$(A + E)\hat{x} = b \tag{1}$$

  where $E$ is a matrix of small perturbations to $A$. The solution to Equation (1) satisfies

$$\frac{\|\hat{x} - x\|}{\|x\|} = \kappa(A)\frac{\|E\|}{\|A\|}$$

  In other words both $\kappa(A)$ and $\|E\|$ affect the accuracy of GEPP.

## 6.3 The role of the residual

- The numerical solution to $Ax = b$ is $\hat{x}$. The residual vector is defined by

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

  Notice that $r$ uses the numerical solution , $\hat{x}$, not the true solution, $x$.

- So, if $r$ is "small enough", is $\hat{x} \approx x$?

  The answer is no!

- It is not hard to show that (see Appendix)

$$\frac{\|\hat{x} - x\|}{\hat{x}} = \kappa(A)\frac{\|r\|}{\|b\|} \tag{2}$$

  Thus, the error

$$\frac{\|\hat{x} - x\|}{x}$$

  is small, *only if*

$$\frac{\|r\|}{\|b\|} \text{ is small}$$

  *and*

$$\kappa(A) \text{ is not large.}$$

## 6.4 Rules of thumb

1 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* $\kappa(A)$ is large.

2 If $A$ and $b$ are stored to machine precision $\varepsilon_m$, the numerical solution to $Ax = b$ by any variant of GEPP is correct to $d$ digits where

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A)) \tag{3}$$

In MATLAB, $\varepsilon_m \approx 2.2 \times 10^{-16}$. Therefore, in MATLAB, there is a maximum of 16 possibly significant digits when solving $Ax = b$. Since $\kappa(A) \geq 1$, we can expect that the solution to linear systems has somewhat fewer than 16 reliable digits.

## 6.5 Summary of Condition Number

- Small residuals are achieved by Gaussian elimination with partial pivoting.)
- Small residuals do not guarantee an accurate solution unless the condition number is "not too large".
- The rule of thumb in Equation (3) provides a way of estimating when condition numbers are "too large"
- In general, we don't compute the condition numbers for routine problem-solving. The MATLAB backslash operator estimates the condition number and prints a warning message if it's too large
- The condition number is a measure of how close the coefficient matrix is to being numerically rank deficient (numerically singular). By "numerical rank deficiency" I mean that a matrix that is not singular in exact arithmetic (infinite number of mantissa digits) can be effectively rank deficient due to roundoff errors.

# Appendix: Derive Equation (2)

Let $x$ be the exact solution and $\hat{x}$ be the numerical solution to $Ax = b$. Define the residual by $r = b - A\hat{x}$, and derive Equation (2).

Add $Ax - b = 0$ to the equation defining the residual, $r = b - A\hat{x}$,

$$r = b - A\hat{x} = b - A\hat{x} + (Ax - b) = A(x - \hat{x})$$

Thus, since $r = A(x - \hat{x})$,

$$x - \hat{x} = A^{-1}r$$

Taking norms of both sides we get

$$\|x - \hat{x}\| \leq \|A^{-1}\|\|r\| \tag{4}$$

Now, take the norm of $Ax = b$

$$\|Ax\| = \|b\| \quad \Longrightarrow \quad \|b\| \leq \|A\|\|x\|$$

and rearrange to get

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

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

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|r\|}{\|b\|}$$

or

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

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