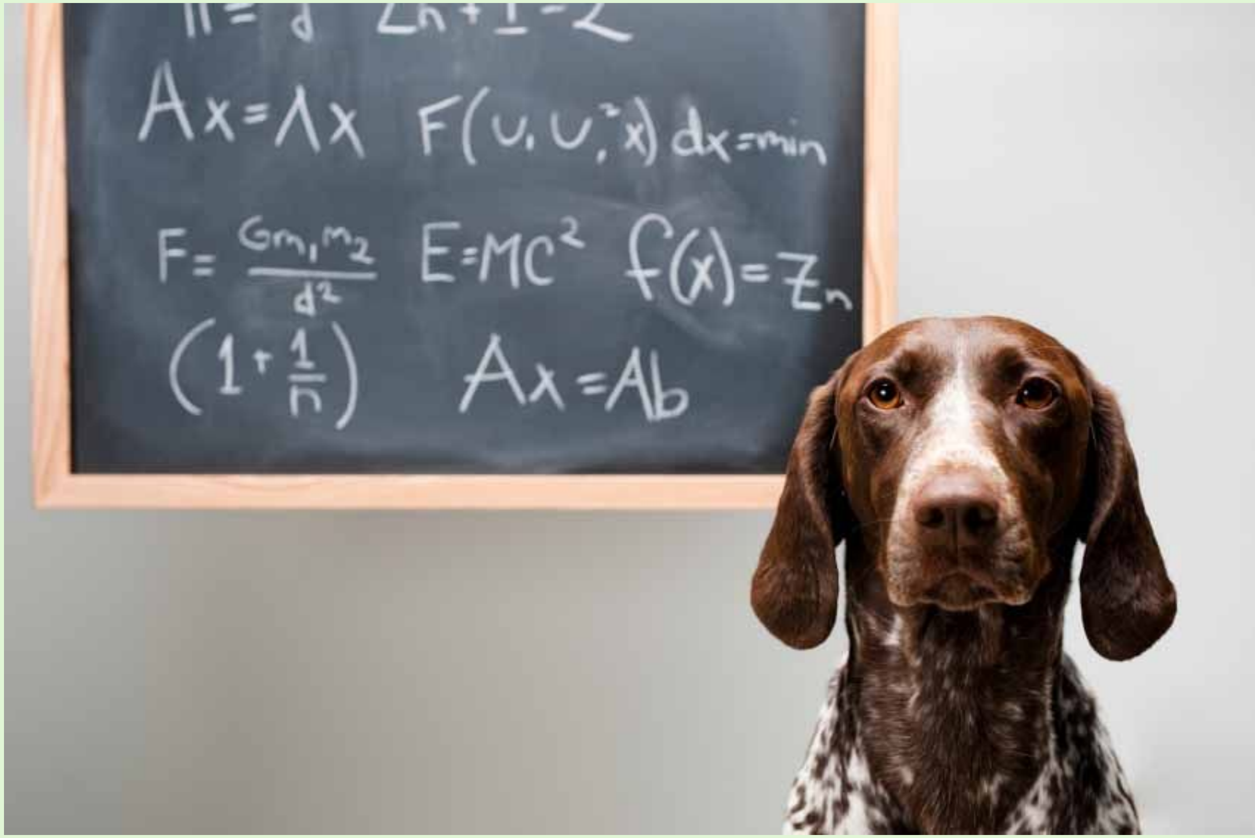
Lecture 18
Linear Algebra, II



Lecture 18 Goals

- Matrix inversion, singularity, rank, and determinants
- Solving systems of linear equations

Problem of inversion

Solve for unknown vector x:

$$Ax = y$$

where A is mxn, x is nx1, and y is mx1.

Problem of inversion

Solve for unknown vector x:

$$Ax = y$$

where A is mxn, x is nx1, and y is mx1.

That's basic algebra!

$$x = y/A$$

Problem of inversion

Solve for unknown vector x:

$$Ax = y$$

where A is mxn, x is nx1, and y is mx1.

Or is it?

$$\cancel{x = y/A}$$

Recall that matrix multiplication is not commutative, so we cannot divide B=C/A since AB/A is not equal to B.

What is Matrix “division” anyway?

Problem of inversion

Solve for unknown vector x:

$$Ax = y$$

where A is mxn, x is nx1, and y is mx1.

Correct solution (use the Matrix inverse):

A\Ax = A\y => x = A\y

Order matters, i.e.

AB = C => A\AB=A\C => B = A\C

Problem of inversion

Solve for unknown vector x:

$$Ax = y$$

where A is mxn, x is nx1, and b is mx1.

Correct solution (use the Matrix inverse):

$$A\backslash Ax = A\backslash y \qquad \Rightarrow \qquad x = A\backslash y$$

For matrices, the definition of the "inverse", or "one over" the matrix, has to be defined properly.

Key question: When does the inverse exist?

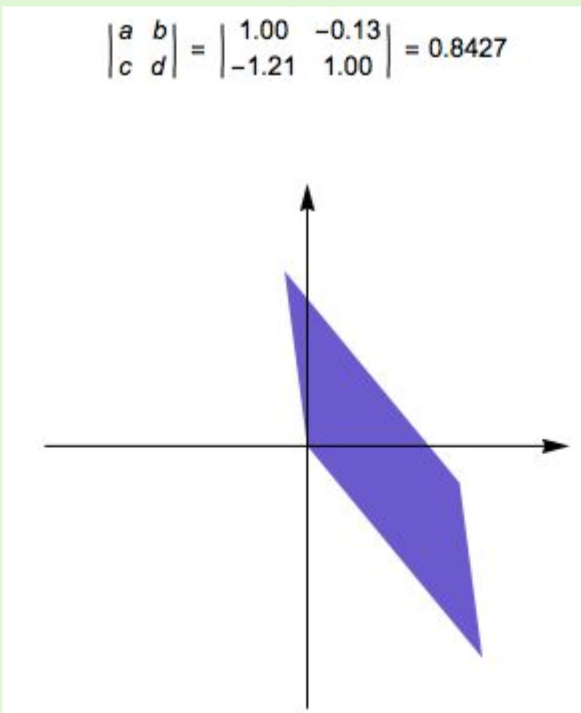
Answer: The Determinant

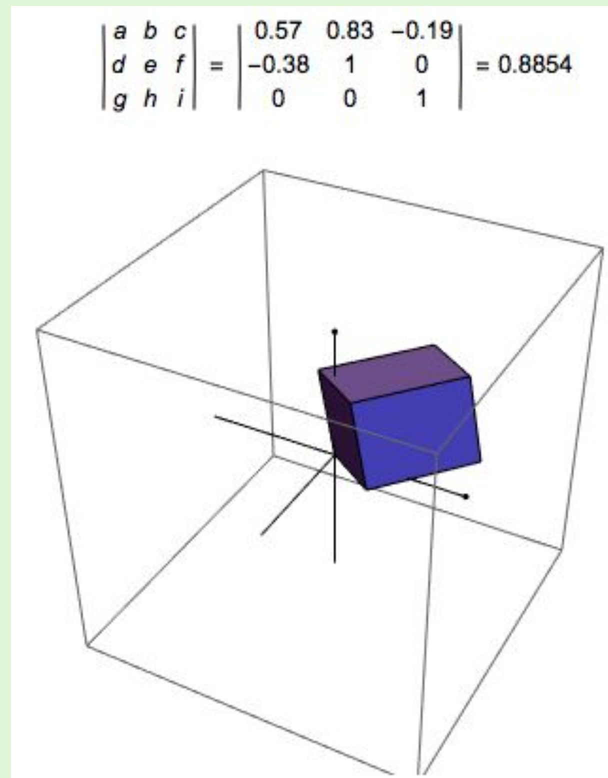
If the determinant is **non-zero**, the matrix can be **inverted** and unique solution exists for Ax=y.

If the determinant is **zero**, the matrix **cannot be inverted**, there can be either 0 or an infinite number of solutions to Ax=y.

Wait. What's the Determinant?

Geometric interpretation: The determinant describes how the area (2D), volume (3D), or hypervolume (4D or higher) defined by set of points X changes when matrix A is applied to X.





Interactive Demonstration: http://demonstrations.wolfram.com/DeterminantsSeenGeometrically/

Determinants (square matrices)

Consider a 2 x 2 matrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

The determinant of a 2 x 2 matrix is

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

Determinants (square matrices)

Finding the determinant of a general n x n square matrix requires evaluation of a complicated polynomial of the coefficients of the matrix, but there is a simple recursive approach.

$$|B| = det(B) = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & k \end{vmatrix} = a\begin{vmatrix} e & f \\ h & k \end{vmatrix} - b\begin{vmatrix} d & f \\ g & k \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

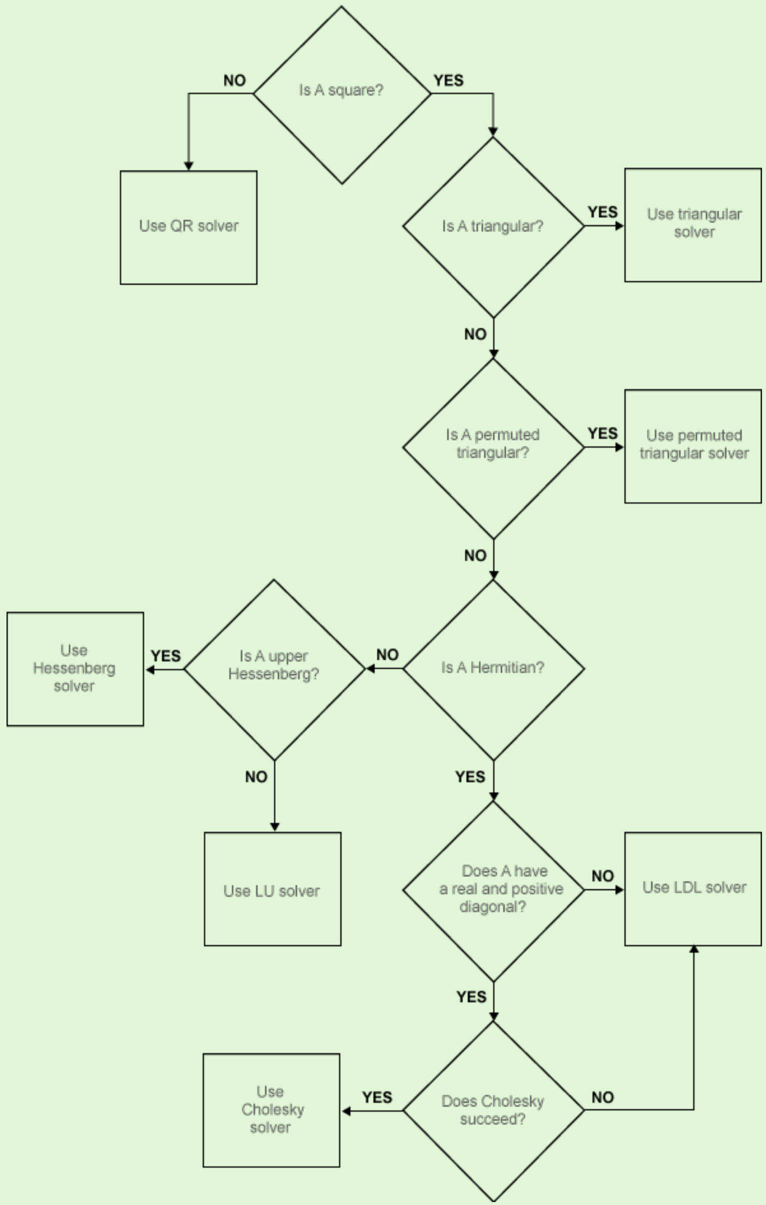
Solving Ax=y

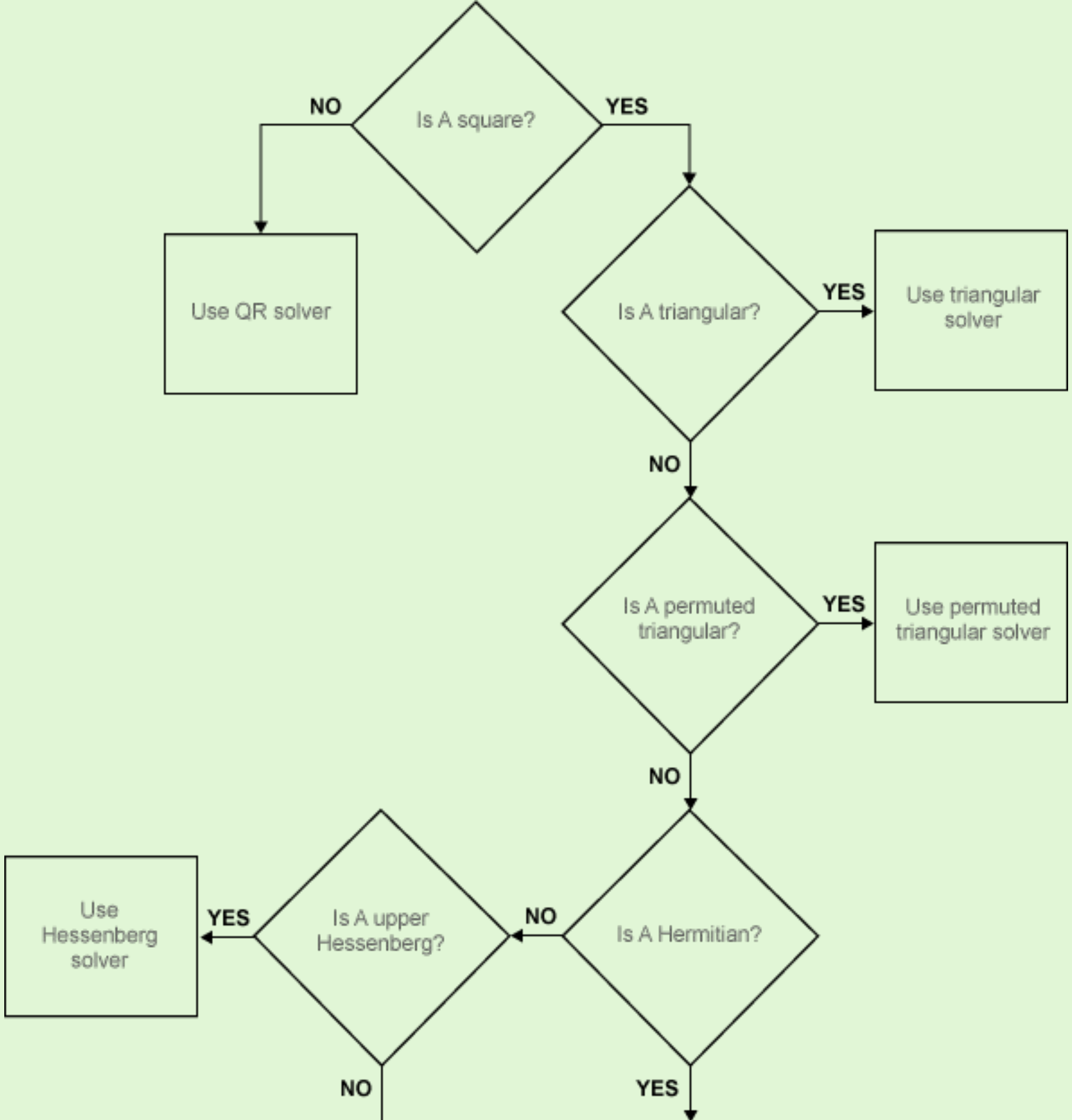
- When A is non-singular (has non-zero determinant) A inverse exists, and one can find x via

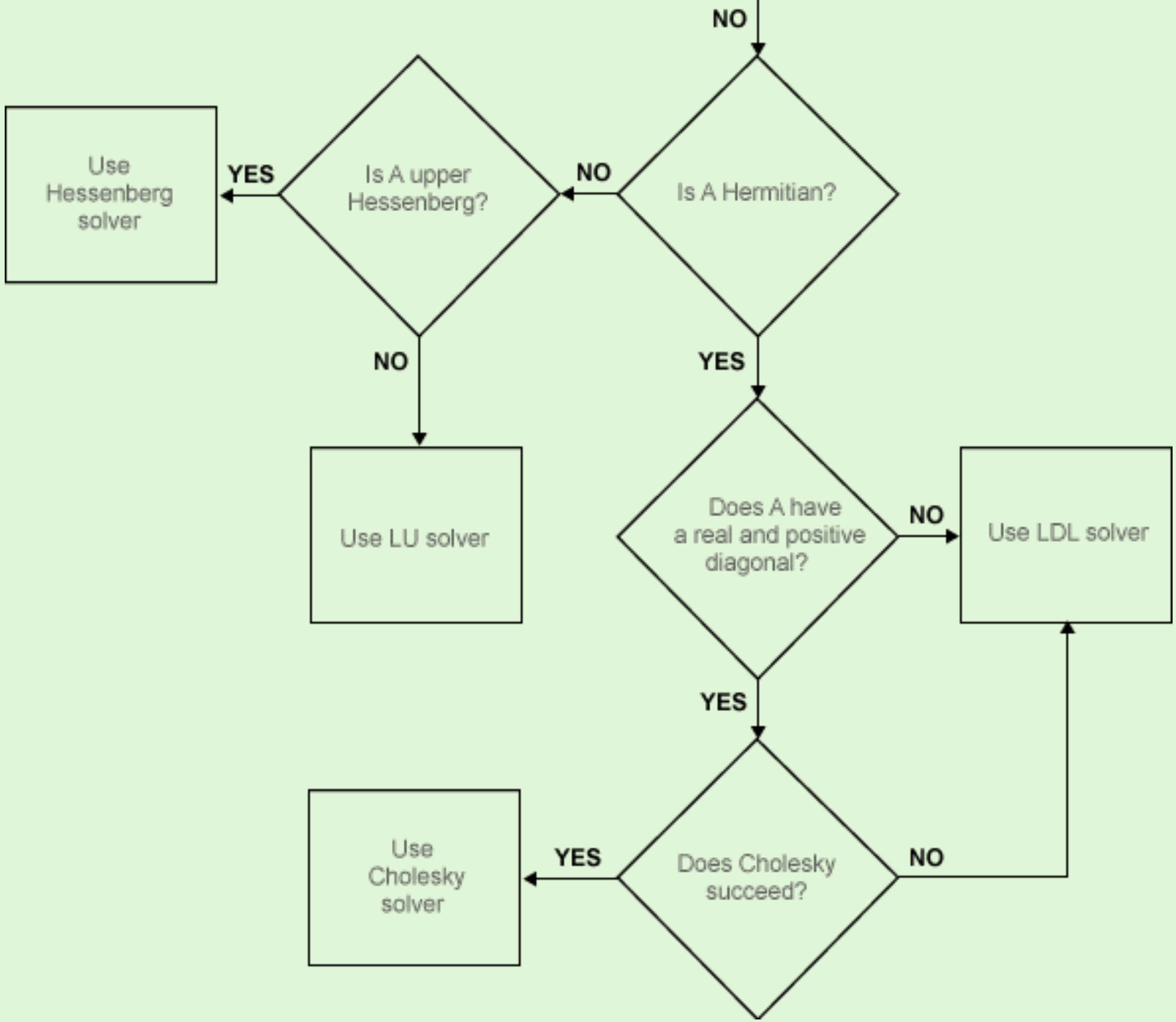
  `x = inv(A)*y`

- However, depending on A, this is can be computationally inefficient and or less precise then using **`x = A\y`**
- The MATLAB \ operation (called mldivide) takes the form of A into account while trying to solve A\y
- `doc mldivide`

Operation of A\y in MATLAB




Is A square?
NO
YES
Use QR solver
Is A triangular?
YES
Use triangular solver
NO
Is A permuted triangular?
YES
Use permuted triangular solver
NO
Is A Hermitian?
NO
Is A upper Hessenberg?
YES
Use Hessenberg solver
NO
YES


NO
Is A Hermitian?
NO
Is A upper Hessenberg?
YES
Use Hessenberg solver
NO
Use LU solver
YES
Does A have a real and positive diagonal?
NO
Use LDL solver
YES
Does Cholesky succeed?
YES
Use Cholesky solver
NO

Linear Equations

- y = mx+b is a linear function, y(x)
- Setting mx + b = c is a linear equation
- Systems of equations

  Have multiple Equations

$$\left\{ \begin{array}{lcr} 2x_1 + 9x_2 & & 5 \\ & = & \\ 3x_1 - 4x_2 & & 7 \end{array} \right.$$

  Solving a system of equations involves finding a set (or sets) of values that allow all the equations to hold.

  These are not always solvable!

Linear equations - Example

Linear equations

$$\begin{cases} 2x_1 + 9x_2 & = & 5 \\ 3x_1 - 4x_2 & & 7 \end{cases}$$

can be written in matrix form

$$\begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

Or more symbolically as

$$y = Ax$$

where

$$A = \begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

Linear equations – General Form

m linear equations with n variables:

$$
\begin{aligned}
y_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\
y_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\
&\vdots \\
y_m &= a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n
\end{aligned}
$$

Can be written in matrix as $y = Ax$ where

$$
y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
$$

Linear equations

For the equation $Ax=y$, there are 3 distinct cases

Square, equal number of unknowns and equations

Underdetermined: more unknowns than equations

Overdetermined: fewer unknowns than equations

Types of solutions with “random” data

“Generally” the following observations would hold

One solution (eg., 2 lines intersect at one point)

Infinite solutions (eg., 2 planes intersect at many points)

No solutions (eg., 3 lines don’t intersect at a point)

Linear equations

But other things can happen.   For example:

No solution (2 parallel lines)
Many solutions (2 parallel lines)

No solutions (2 parallel planes)

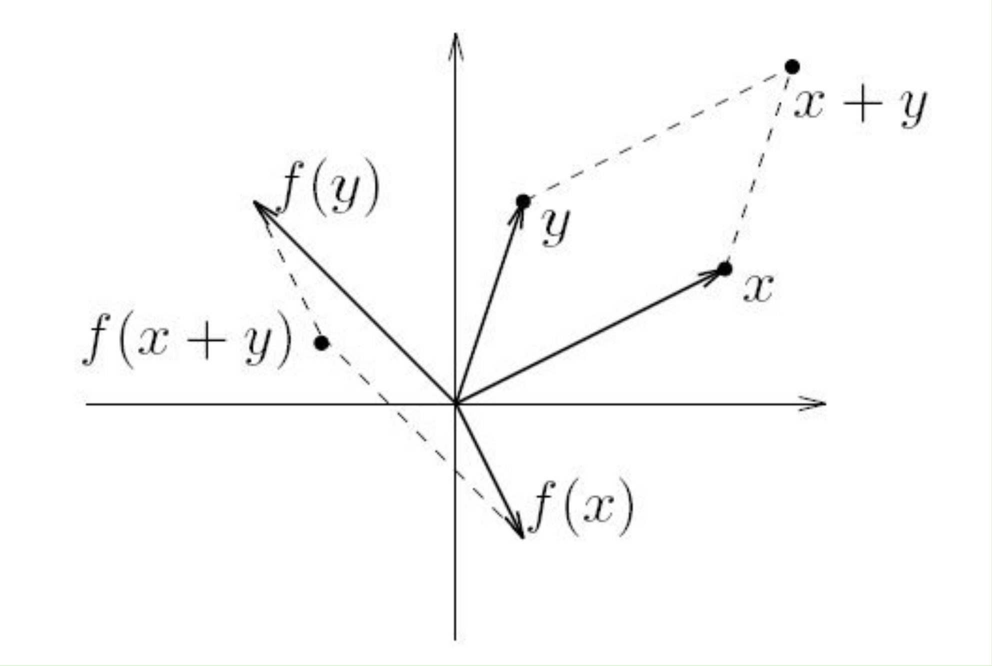
Solutions (3 lines that do intersect at a point)

What is a linear function?

Let $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ be a function. It is said to be linear if

- $f(x+y) = f(x) + f(y),\ \forall x, y \in \mathbf{R}^n$
- $f(\alpha x) = \alpha f(x),\ \forall x \in \mathbf{R}^n\ \forall \alpha \in \mathbf{R}$

Superposition can be applied to linear systems!



Matrix representation of a linear function

- consider function $f : \mathbf{R}^n \to \mathbf{R}^m$ given by $f(x) = Ax$, where $A \in \mathbf{R}^{m \times n}$
- matrix multiplication function f is linear
- **converse** is true: **any** linear function $f : \mathbf{R}^n \to \mathbf{R}^m$ can be written as $f(x) = Ax$ for some $A \in \mathbf{R}^{m \times n}$
- representation via matrix multiplication is unique: for any linear function f there is only one matrix A for which $f(x) = Ax$ for all x
- $y = Ax$ is a concrete representation of a generic linear function

What is a linear function?

Let $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ unction. It is said to be linear if

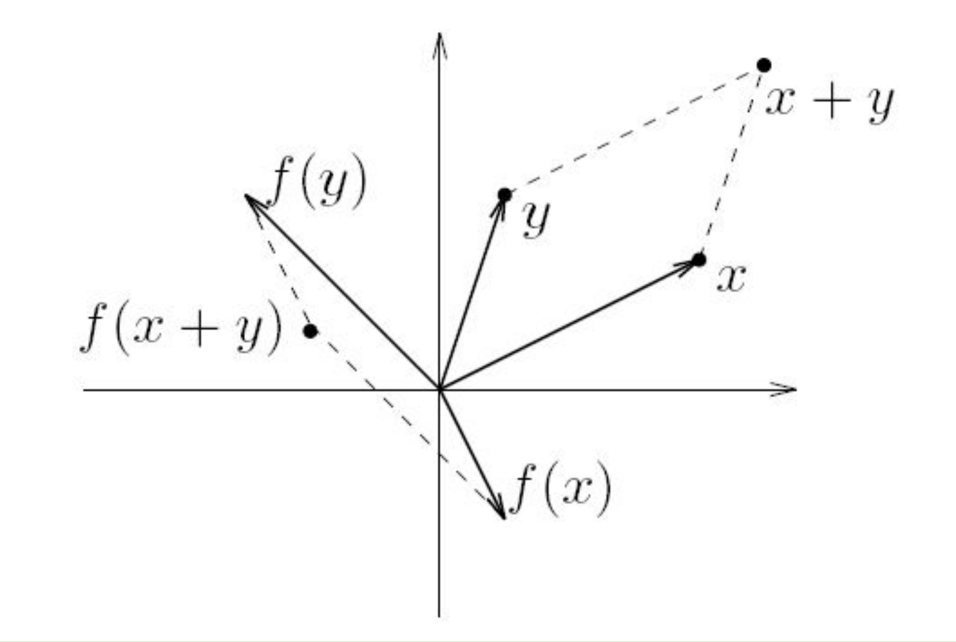
- $f(x+y) = f(x) + f(y), \ \forall x, y \in \mathbf{R}^n$
- $f(\alpha x) = \alpha f(x), \ \forall x \in \mathbf{R}^n \ \forall \alpha \in \mathbf{R}$

This is sometimes called superposition

When f is represented with matrix A, it is clear it satisfies the properties above, i.e.,

A(x+y) = Ax+Ay

A(cx) = cAx



Solving Ax=b

Two key questions

1) How can we tell when no solutions exist?

2) When solutions exist how can we find and represent them?

Range of a matrix

the *range* of $A \in \mathbf{R}^{m \times n}$ is defined as

$$\mathcal{R}(A) = \{Ax \mid x \in \mathbf{R}^n\} \subseteq \mathbf{R}^m$$

$\mathcal{R}(A)$ can be interpreted as

- the set of vectors that can be 'hit' by linear mapping $y = Ax$
- the span of columns of A
- the set of vectors y for which $Ax = y$ has a solution

Ax=b has a solution when b is in R(A)

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1m} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1m}x_m \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2m}x_m \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nm}x_m \end{bmatrix}$$

$$\begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{n1} \end{bmatrix} [x_1] + \begin{bmatrix} A_{12} \\ A_{22} \\ \vdots \\ A_{n2} \end{bmatrix} [x_2] + \begin{bmatrix} A_{13} \\ A_{23} \\ \vdots \\ A_{n3} \end{bmatrix} [x_3] + \cdots + \begin{bmatrix} A_{1m} \\ A_{2m} \\ \vdots \\ A_{nm} \end{bmatrix} [x_m]$$

Linear independence

a set of vectors $\{v_1, v_2, \ldots, v_k\}$ is *independent* if

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = 0 \Longrightarrow \alpha_1 = \alpha_2 = \cdots = 0$$

some equivalent conditions:

- coefficients of $\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k$ are uniquely determined, *i.e.*,

  $$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = \beta_1 v_1 + \beta_2 v_2 + \cdots + \beta_k v_k$$

  implies $\alpha_1 = \beta_1, \alpha_2 = \beta_2, \ldots, \alpha_k = \beta_k$

- no vector v_i can be expressed as a linear combination of the other vectors $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k$

Rank of a matrix

The rank of a matrix is equal to the maximal number of linearly independent vectors in the range of the matrix.

A square matrix A of dimension n x n is said to be **full rank** if the rank of the matrix is n, i.e. **rank(A) = n**

If a matrix is full rank, it can be inverted.

The rank of a matrix can be computed using the command rank

```
A =

     1     1     1     1
     1    -2    -3    -4
     1     2     3     4
     1     2     2     1
```

```
>> rankOfA=rank(A)

rankOfA =

     4
```

Key Question #1: Existence

How can we tell when no solution exists to Ax=b?

- When rank([A b]) > rank(A) there isn't a linear combination of the columns of A that can be used to represent y, i.e. Ax = b has no solution.

Key Question #2: Uniqueness?

How can we tell the solution to $Ax=b$ is unique?

When multiple solutions exists how can we find and represent them?

Nullspace of a matrix

the *nullspace* of $A \in \mathbf{R}^{m \times n}$ is defined as

$$\mathcal{N}(A) = \{ \, x \in \mathbf{R}^n \mid Ax = 0 \, \}$$

- $\mathcal{N}(A)$ is set of vectors mapped to zero by $y = Ax$
- $\mathcal{N}(A)$ is set of vectors orthogonal to all rows of A

Nullspace != {0} => Infinite Solutions

the *nullspace* of $A \in \mathbf{R}^{m \times n}$ is defined as

$$\mathcal{N}(A) = \{ \, x \in \mathbf{R}^n \mid Ax = 0 \, \}$$

- $\mathcal{N}(A)$ is set of vectors mapped to zero by $y = Ax$
- $\mathcal{N}(A)$ is set of vectors orthogonal to all rows of A

$\mathcal{N}(A)$ gives *ambiguity* in x given $y = Ax$:

- if $y = Ax$ and $z \in \mathcal{N}(A)$, then $y = A(x + z)$
- conversely, if $y = Ax$ and $y = A\tilde{x}$, then $\tilde{x} = x + z$ for some $z \in \mathcal{N}(A)$

Existence and Uniqueness

- Ax=b
- Existence: Having a solution means
  b in R(A)
- Having a unique solution means Az=0 iff z is an appropriately sized zero vector
  - Otherwise x+z != x, and A(x+z)=Ax+Az = Ax = b
- Uniqueness: Having a unique solution requires
  N(A)={0}

When do we need to compute N(A)?

For dim(A) m by n,

dim(Null(A)) is m by n-rank(A).

So when A is full rank (i.e. rank(A) = n), N(A) = {0}

Finding the nullspace in MATLAB

null(A) returns an orthonormal basis for the null space of A
null(A,’r’) returns a “rational” basis for the null space of A

For illustrative purposes, use the script initializeMatrices.m, which puts 4 matrices of size 4 x 4 into the workspace:

A is of rank 4
B is of rank 3
C is of rank 2
D is or rank 1

```
>> initializeMatrices
>> whos
  Name      Size                Bytes  Class

  A         4x4                   128  double array
  B         4x4                   128  double array
  C         4x4                   128  double array
  D         4x4                   128  double array

Grand total is 64 elements using 512 bytes
```

Finding the null space in MATLAB

Since A is of rank 4, the null space of A is reduced to zero.

The null space of the matrix B (of rank 3) is of dimension 1

```
>> nullA=null(A,'r')

nullA =

  Empty matrix: 4-by-0

>> nullB=null(B,'r')

nullB =

   -1
    3
   -3
    1
```

The command null(A,’r’) will take the matrix A and return the set of vectors which belong to the null space of A. The ‘r’ is added so Matlab returns fractional vectors when possible.

Null space of A is empty, because the matrix is invertible

Null space of B is of dimension 1

Sanity checks

```
>> initializeMatrices
>> whos
  Name      Size                    Bytes  Class

  A         4x4                       128  double array
  B         4x4                       128  double array
  C         4x4                       128  double array
  D         4x4                       128  double array

Grand total is 64 elements using 512 bytes
```

```
>> nullB = null(B,'r')

nullB =

    -1
     3
    -3
     1

>> B*nullB

ans =

     0
     0
     0
     0
```

Sanity checks

```
>> initializeMatrices
>> whos
  Name      Size                    Bytes  Class

  A         4x4                       128  double array
  B         4x4                       128  double array
  C         4x4                       128  double array
  D         4x4                       128  double array

Grand total is 64 elements using 512 bytes
```

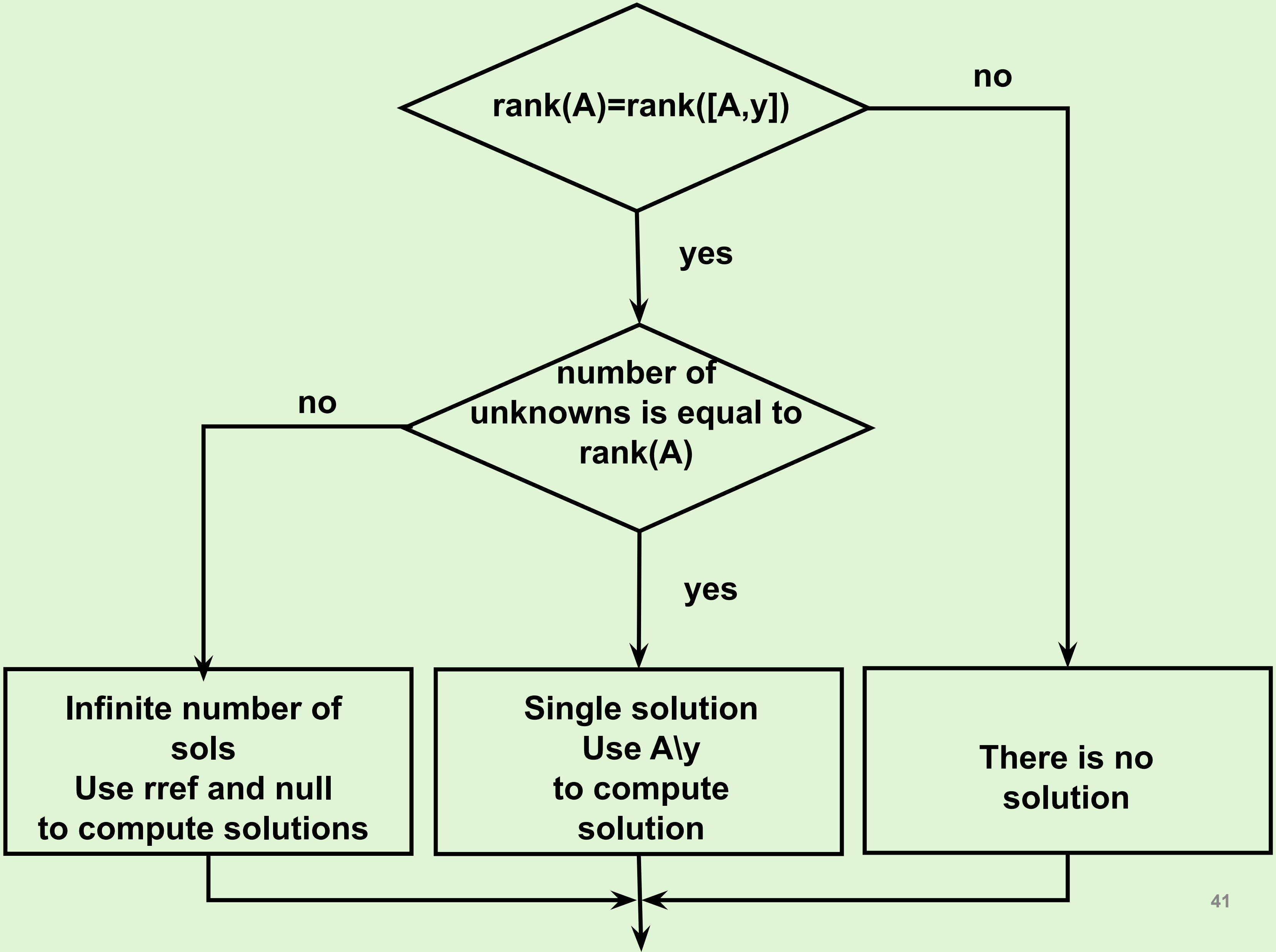
```
>> nullC = null(C,'r')

nullC =

     0    -1
    -1     0
     1     0
     0     1

>> C*nullC

ans =

     0     0
     0     0
     0     0
     0     0
```

Sanity checks

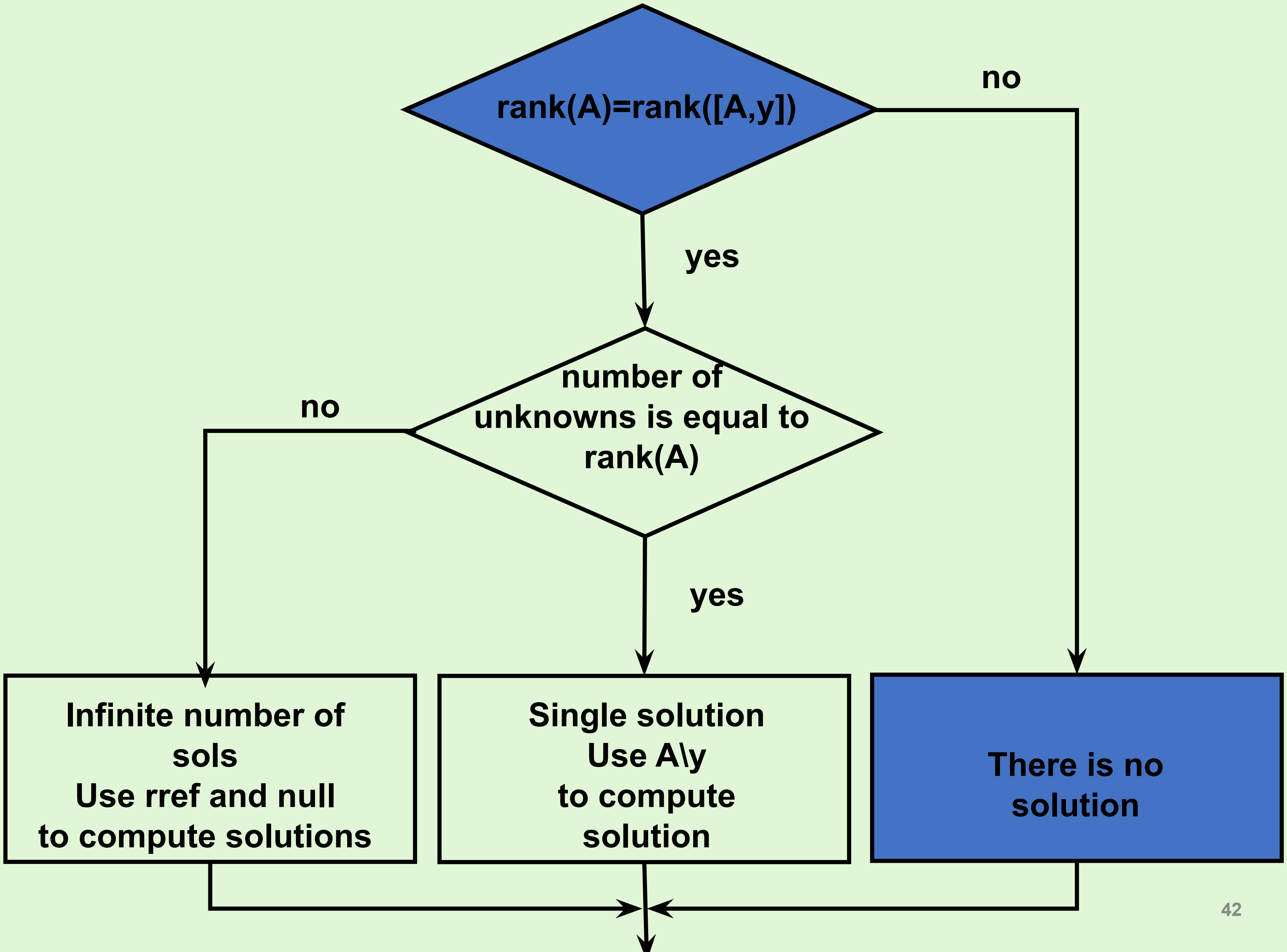
```
>> initializeMatrices
>> whos
  Name      Size                    Bytes  Class

  A         4x4                       128  double array
  B         4x4                       128  double array
  C         4x4                       128  double array
  D         4x4                       128  double array

Grand total is 64 elements using 512 bytes
```

```
>> nullD = null(D,'r')

nullD =

    -1    -1    -1
     1     0     0
     0     1     0
     0     0     1

>> D*nullD

ans =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

Procedure to solve a linear system



Procedure to solve a linear system



Step: check if rank(A) = rank([A,y]) → no

```
>> A = [1,2,3;0,3,1;1,14,7]

A =

     1     2     3
     0     3     1
     1    14     7

>> y=[1;2;3]

y =

     1
     2
     3
```

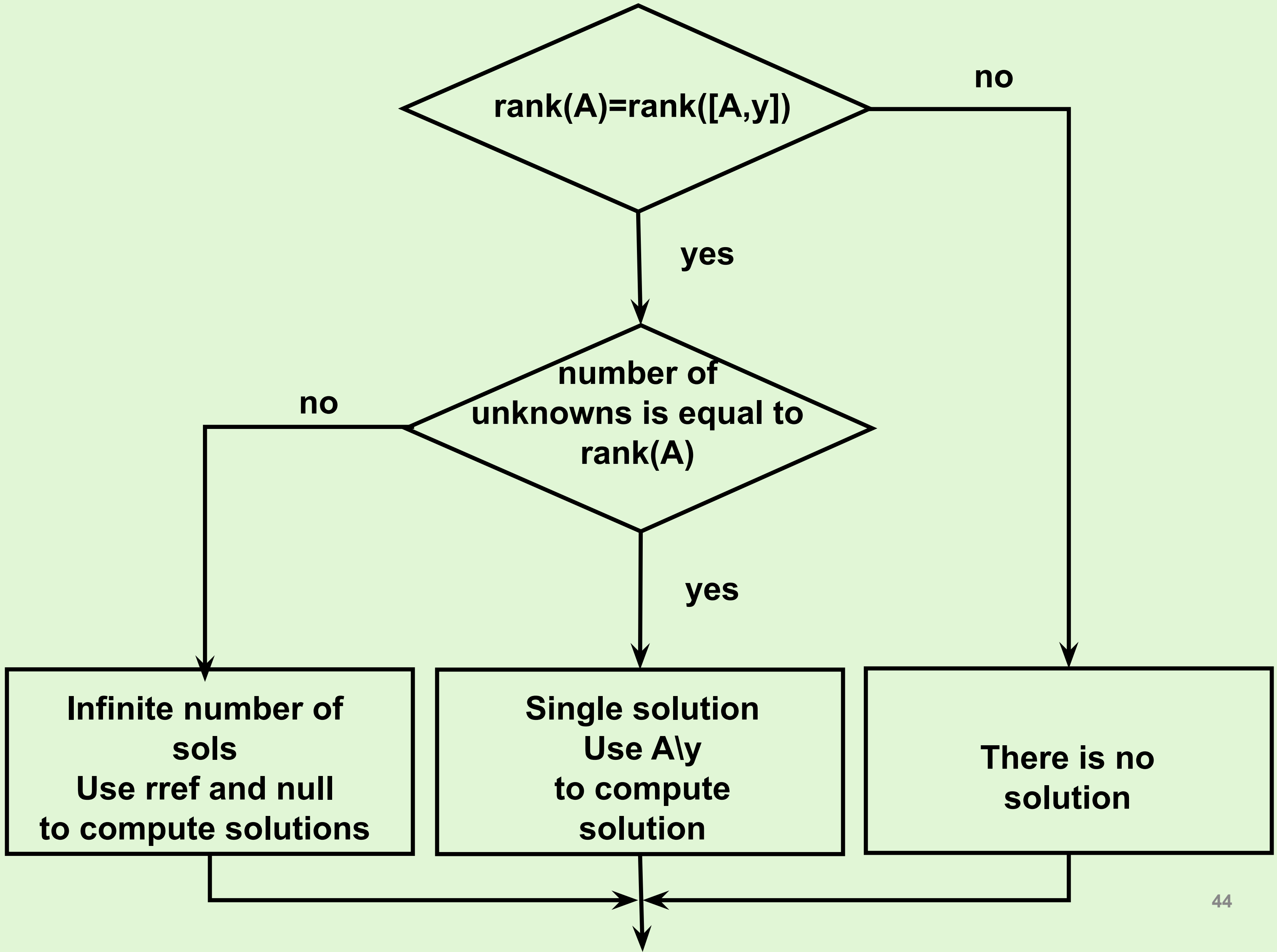
```
>> rank(A)

ans =

     2

>> rank([A,y])

ans =

     3
```

If rank(A) is not equal to rank([A,y]), y is not in the range of A

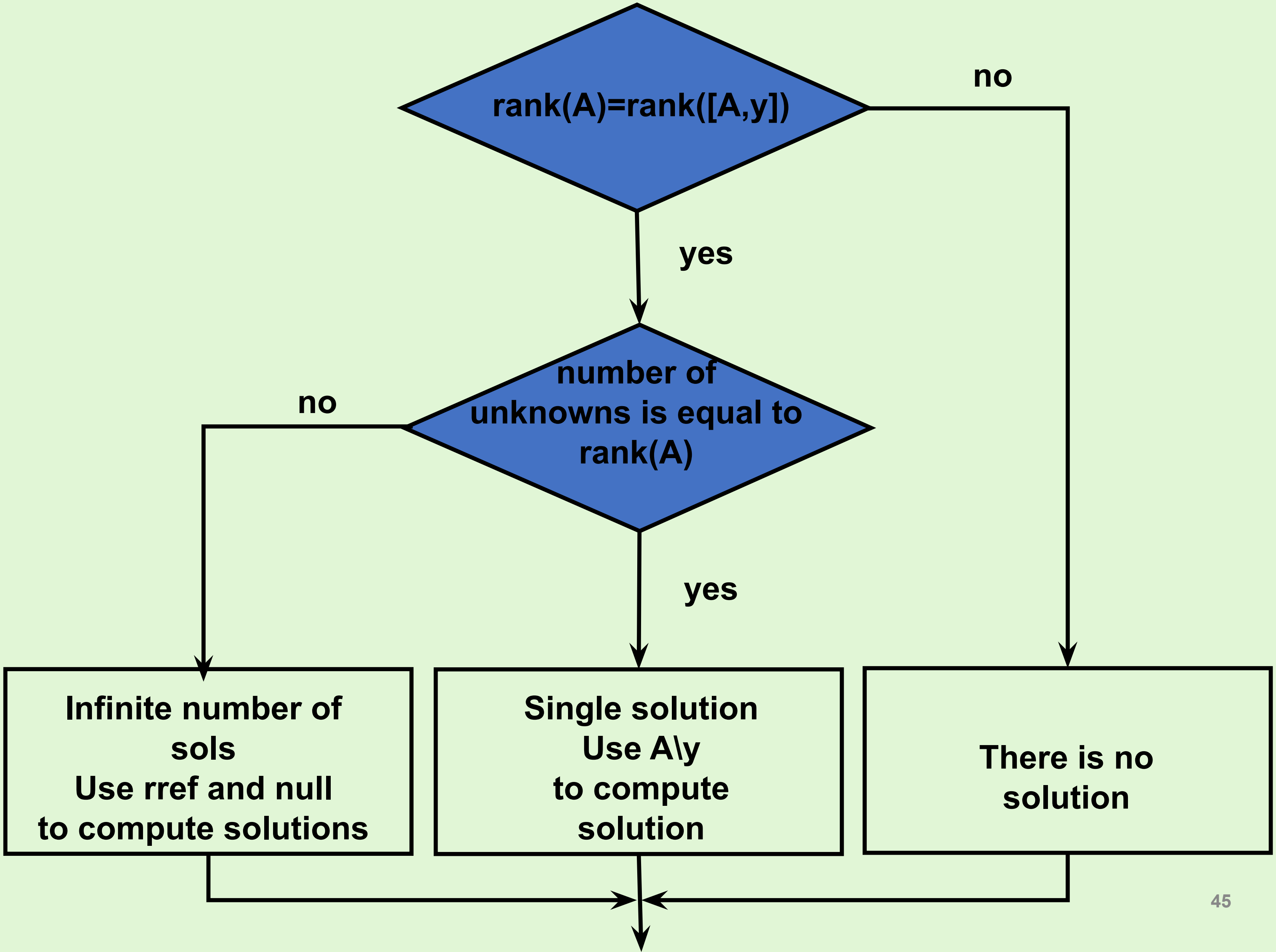
i.e. There is no x such that Ax = y

System has no solution

Procedure to solve a linear system



Procedure to solve a linear system



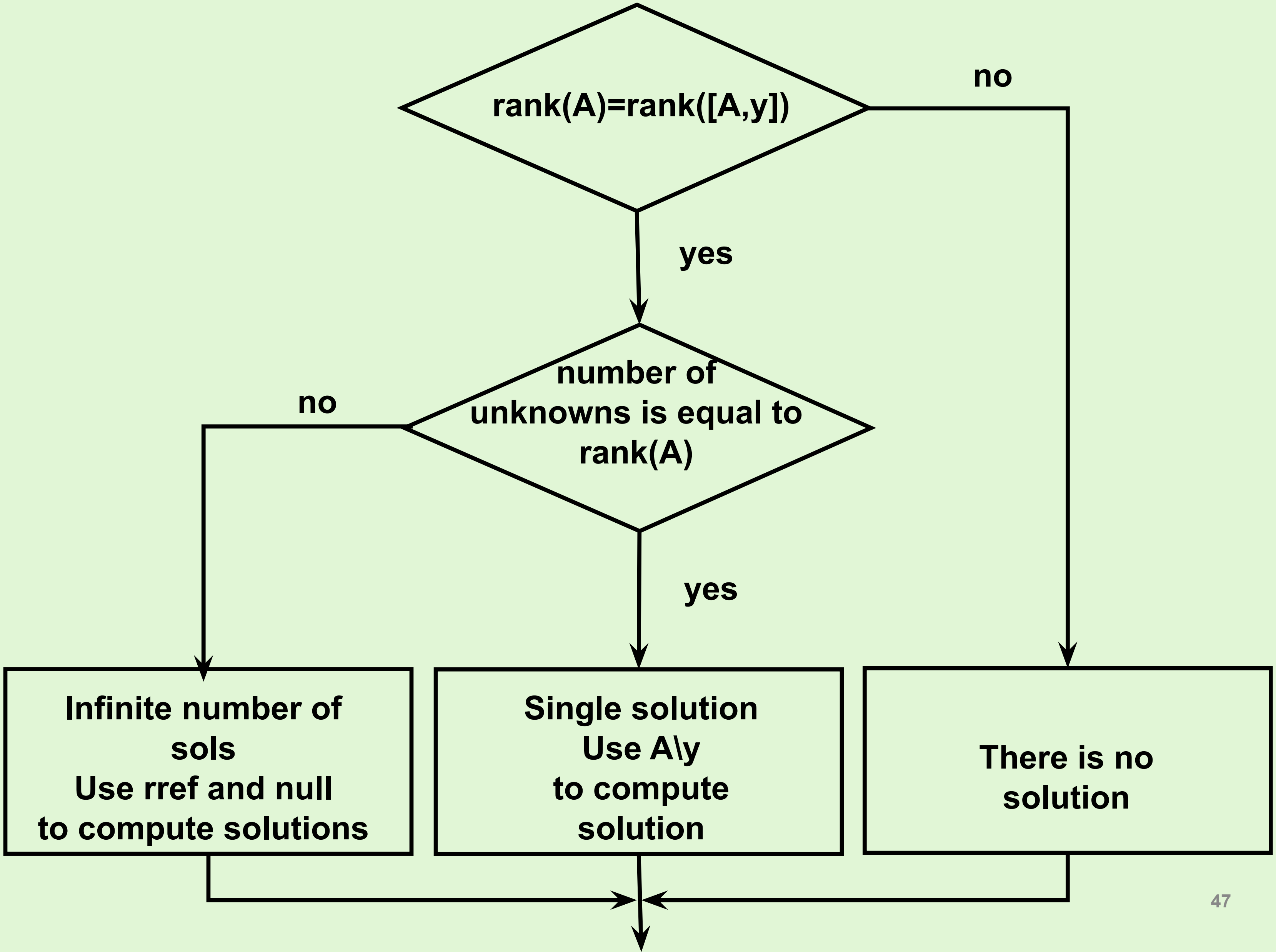
Step: rank(A) = number of unknowns? → yes

```
>> A = [1,2,3;0,3,1;-1,14,7]

A =

     1     2     3
     0     3     1
    -1    14     7

>> rank(A)

ans =

     3

>> y=[1;2;3]

y =

     1
     2
     3
```

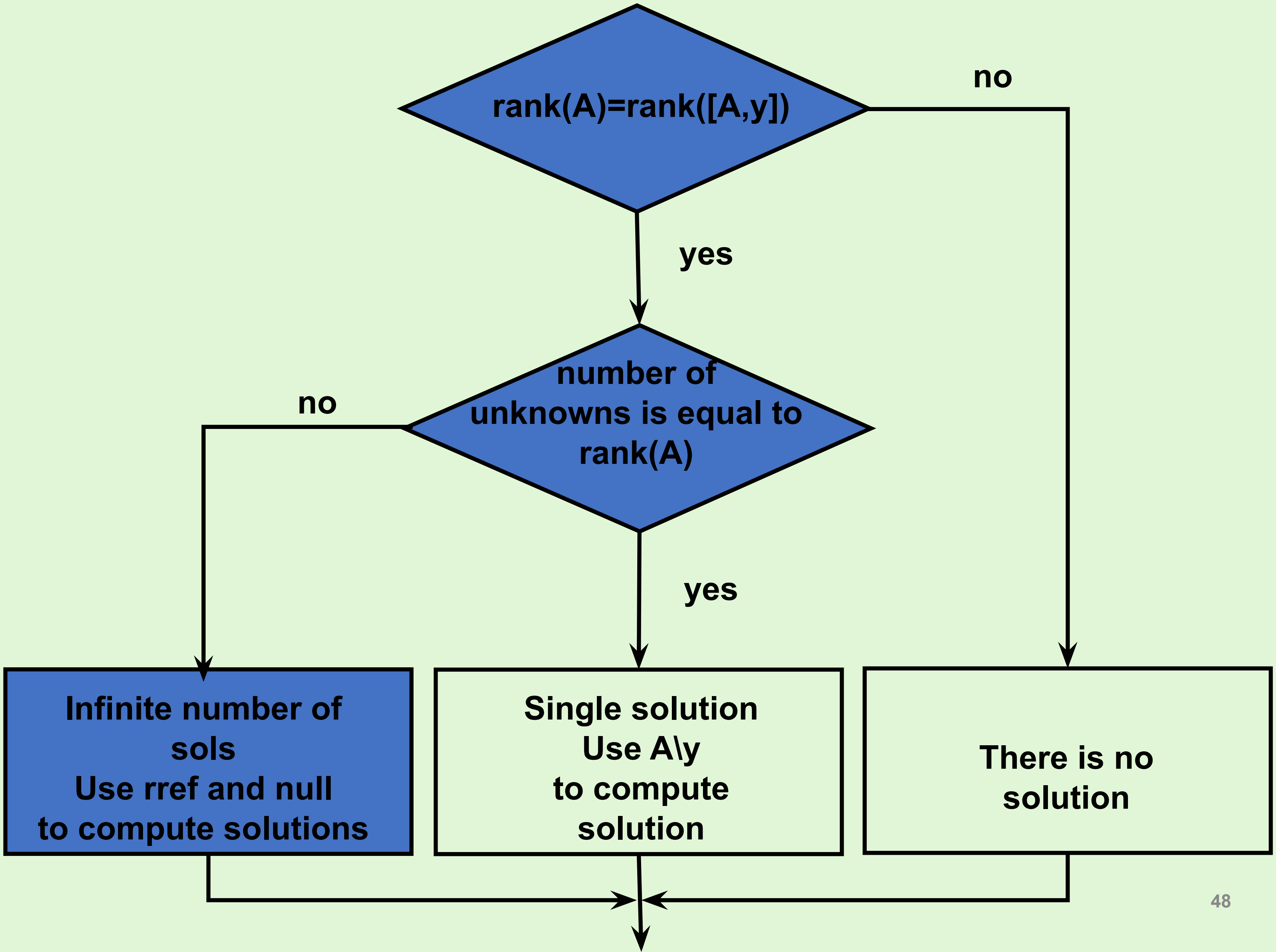
```
>> x=A\y

x =

    3.0000
    1.1429
   -1.4286
```

If the rank of A is the same as the number of unknowns, the system can be inverted, and the system has a unique solution, which can be computed by A\y

Procedure to solve a linear system



Procedure to solve a linear system



Step: rank(A) = number of unknowns? → no

In this case rank(A) < number of unknowns (i.e. it is smaller). An infinite number of solutions exist.

Imagine you can find a particular solution to this problem

$$y = Ax_{particular}$$

Then if you take any vector in the null space, it is also a solution to this problem:

$$y = A(x_{particular} + v_{null\ space})$$

Because

$$Av_{null\ space} = 0$$

Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

$$y = A(x_{particular} + v_1)$$
$$y = A(x_{particular} + 2v_1)$$
$$y = A(x_{particular} + 3v_1)$$
$$y = A(x_{particular} + \lambda_1 v_1)$$

Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

$$y = A(x_{particular} + v_1 + v_2)$$
$$y = A(x_{particular} + 2v_1 + 2v_2)$$
$$y = A(x_{particular} + 3v_1 + 3v_2)$$
$$y = A(x_{particular} + \lambda_1 v_1 + \lambda_2 v_2)$$

Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

$$y = A(x_{particular} + v_1 + v_2 + v_3)$$
$$y = A(x_{particular} + 2v_1 + 2v_2 + 2v_3)$$
$$y = A(x_{particular} + 3v_1 + 3v_2 + 3v_3)$$
$$y = A(x_{particular} + \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3)$$

Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

$$
\begin{aligned}
y &= A(x_{particular} + v_1 + v_2 + v_3 + \cdots + v_k) \\
y &= A(x_{particular} + 2v_1 + 2v_2 + 2v_3 + \cdots + 2v_k) \\
y &= A(x_{particular} + 3v_1 + 3v_2 + 3v_3 + \cdots + 3v_k) \\
y &= A(x_{particular} + \underbrace{\lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3 + \cdots + \lambda_k v_k}_{\text{How far can you go?}})
\end{aligned}
$$

You can go as far as the null space permits, i.e. pick v's from columns of null spaces. Their linear combinations span the null space.

Step: rank(A) = number of unknowns? → no

How to do this with matlab?

Use the command rref

```
>> A

A =

     1     2     3
     0     3     1
     1    14     7

>> rank(A)

ans =

     2
```

```
>> y

y =

     1
     3
    13

>> rank([A,y])

ans =

     2
```

y is in the range of A.
There is an infinity of solutions

How to compute them

For this, you need to use

B = rref([A,y])

Step: rank(A) = number of unknowns? → no

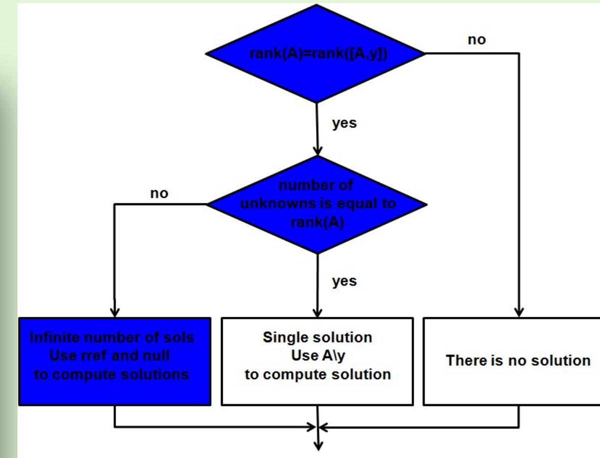
```
>> B=rref([A,y])

B =

   1.0000         0    2.3333   -1.0000
        0    1.0000    0.3333    1.0000
        0         0         0   0
```

Gauss form of A

Particular solution

Find a solution of this system (Gauss elimination has already been done for you)

Step: rank(A) = num. of unk.? → no




```
>> B=rref([A,y])

B =

   1.0000         0    2.3333   -1.0000
        0    1.0000    0.3333    1.0000
        0         0         0         0
```

Obvious solution: [-1;1;0]

```
>> A

A =

     1     2     3
     0     3     1
     1    14     7

>> A*[-1;1;0]

ans =

     1
     3
    13
```

Find a solution of this system (Gauss elimination has already been done for you)

Step: rank(A) = number of unknowns? → no

How do I find all solutions?

Add all the vectors from
the null space

```
vNull =


   -0.9113
   -0.1302
    0.3906

>> vParticular = [-1;1;0]

vParticular =


    -1
     1
     0
```

Step: rank(A) = number of unknowns? → no

How do I find all solutions?

Add all the vectors from the null space

```
vNull =

   -0.9113
   -0.1302
    0.3906

>> vParticular = [-1;1;0]

vParticular =

    -1
     1
     0
```

```
>> vParticular = [-1;1;0]

vParticular =

    -1
     1
     0

>> A*(vParticular+vNull)

ans =

    1.0000
    3.0000
   13.0000
```

```
>> A*(vParticular+2*vNull)

ans =

    1.0000
    3.0000
   13.0000
```

Gauss Elimination (non-singular A)

Want to solve Ax = b

- Forward elimination
  - Starting with the first row, add or subtract multiples of that row to eliminate the first coefficient from the second row and beyond.
  - Continue this process with the second row to remove the second coefficient from the third row and beyond.
  - Stop when an upper triangular matrix remains.
- Back substitution
  - Starting with the *last* row, solve for the unknown, then substitute that value into the next highest row.
  - Because of the upper-triangular nature of the matrix, each row will contain only one more unknown.

(*a*) Forward elimination

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right] \downarrow \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array}\right]$$

(*b*) Back substitution

$$x_3 = b''_3/a''_{33}$$

$$x_2 = (b'_2 - a'_{23}x_3)/a'_{22}$$

$$x_1 = (b_1 - a_{13}x_3 - a_{12}x_2)/a_{11}$$

Order of Elimination

$$\begin{matrix} ? & ? & ? & ? \\ 1 & ? & ? & ? \\ 2 & 4 & ? & ? \\ 3 & 5 & 6 & ? \end{matrix} = \begin{matrix} ? \\ ? \\ ? \\ ? \end{matrix}$$

Gaussian Elimination in 3D

$$\begin{aligned} 2x &+ 4y - 2z = 2 \\ 4x &+ 9y - 3z = 8 \\ -2x &- 3y + 7z = 10 \end{aligned}$$

- Using the first equation to eliminate *x* from the next two equations

Gaussian Elimination in 3D

$$\begin{aligned} 2x + 4y - 2z &= 2 \\ y + z &= 4 \\ y + 5z &= 12 \end{aligned}$$

- Using the second equation to eliminate *y* from the third equation

Gaussian Elimination in 3D

$$\begin{aligned} 2x + 4y - 2z &= 2 \\ y + z &= 4 \\ 4z &= 8 \end{aligned}$$

- Using the second equation to eliminate *y* from the third equation

Solving Triangular Systems

- We now have a triangular system which is easily solved using a technique called ***Backward-Substitution***.

$$\begin{aligned} 2x + 4y - 2z &= 2 \\ y + z &= 4 \\ 4z &= 8 \end{aligned}$$

Solving Triangular Systems

- If A is upper triangular, we can solve $Ax = b$ by:

$$x_n = b_n / A_{nn}$$

$$x_i = \left(b_i - \sum_{j=i+1}^{n} A_{ij} x_j \right) / A_{ii}, \quad i = n-1, \ldots, 1$$

Backward Substitution

- From the previous work, we have

$$\begin{aligned} 2x + 4y - 2z &= 2 \\ y + z &= 4 \\ z &= 2 \end{aligned}$$

- And substitute z in the first two equations

Backward Substitution

$$\begin{aligned} 2x + 4y - 4 &= 2 \\ y + 2 &= 4 \\ z &= 2 \end{aligned}$$

- We can solve *y*

Backward Substitution

$$\begin{aligned} 2x + 4y - 4 &= 2 \\ y &= 2 \\ z &= 2 \end{aligned}$$

- Substitute to the first equation

Gauss-Jordan Elimination

Keep going until augmented matrix is **reduced row echelon form** (rref):

1. The rows (if any) consisting entirely of zeros are grouped together at the bottom of the matrix.
2. In each row that does not consist entirely of zeros, the leftmost nonzero element is a 1 (called a leading 1 or a pivot).
3. Each column that contains a leading 1 has zeros in all other entries.
4. The leading 1 in any row is to the left of any leading 1's in the rows below it.

Stop process in step 2 if you obtain a row whose elements are all zeros except the last one on the right. In that case, the system is inconsistent and has no solutions. Otherwise, finish step 2 and read the solutions of the system from the final matrix

Overdetermined systems

consider $y = Ax$ where $A \in \mathbf{R}^{m \times n}$ is (strictly) skinny, $i.e.$, $m > n$

- called *overdetermined* set of linear equations (more equations than unknowns)
- for most y, cannot solve for x

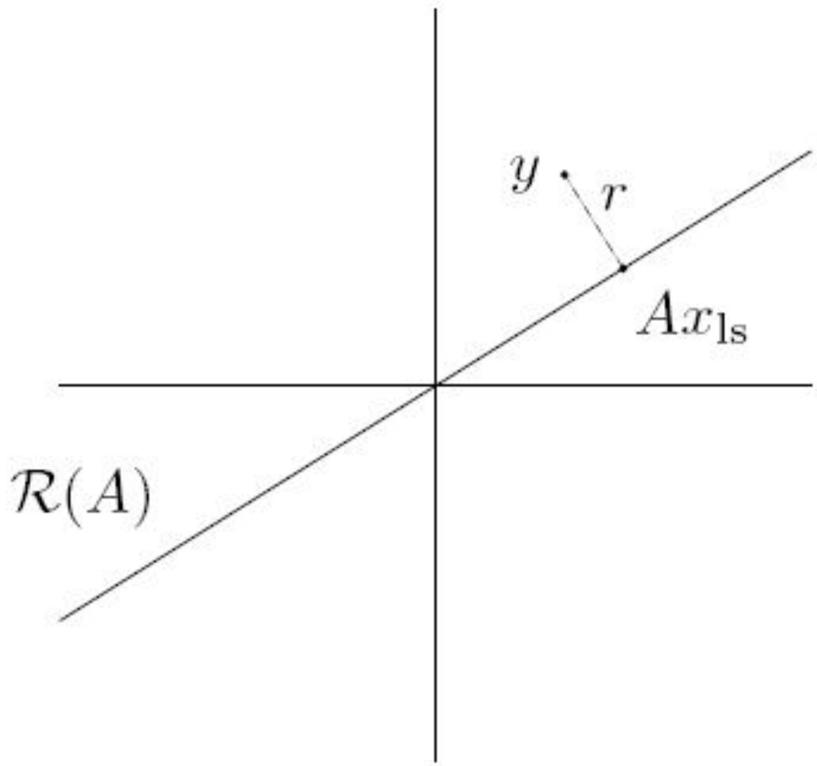
one approach to *approximately* solve $y = Ax$:

- define *residual* or error $r = Ax - y$
- find $x = x_{\mathrm{ls}}$ that minimizes $\|r\|$

x_{ls} called *least-squares* (approximate) solution of $y = Ax$

Overdetermined systems

Ax_{ls} is point in $\mathcal{R}(A)$ closest to y (Ax_{ls} is *projection* of y onto $\mathcal{R}(A)$)



The “Least Squares” Problem

If A is an *n-by-m* array, and b is an *n-by-1* vector, let c^* be the smallest possible (over <u>all choices</u> of *m-by-1* vectors x) mismatch between Ax and b (ie., pick x to make Ax as much like b as possible).

$$c^* \underbrace{:=}_{\text{“is defined as”}} \min_{x,\, m\times 1} \|Ax - b\|$$

“the minimum, over all *m-by-1* vectors *x*”

“the length (ie., norm) of the difference/mismatch between *Ax* and *b*.”

Four cases for Least Squares

Recall least squares formulation

$$c^* := \min_{x, m\times 1} \left\| Ax - b \right\|$$

There are 4 scenarios

$c^* = 0$: the equation $Ax=b$ has <u>at least one</u> solution

- <u>only one</u> x vector achieves this minimum
- <u>many different</u> x vectors achieves the minimum

$c^* > 0$: the equation $Ax=b$ has <u>no</u> solutions

- <u>only one</u> x vector achieves this minimum
- <u>many different</u> x vectors achieves the minimum

Four cases: **`x=A\b`** as solution of $c^* := \min_{x,m\times 1} \|Ax-b\|$

No Mismatch:

$c^* = 0$, and <u>only one</u> *x* vector achieves this minimum

Choose this *x*

$c^* = 0$, and <u>many different</u> *x* vectors achieves the minimum

From all these minimizers, choose smallest *x* (ie., norm)

Mismatch:

$c^* > 0$, and <u>only one</u> *x* vector achieves this minimum

Choose this *x*

$c^* > 0$, and <u>many different</u> *x* vectors achieves the minimum

From all minimizers, choose an *x* with the smallest norm

The backslash operator

If A is an *n-by-m* array, and b is an *n-by-1* vector, then

```
>> x = A\b
```

solves the "least squares" problem. Namely

- If there is an x which solves $Ax=b$, then this x is computed
- If there is no x which solves $Ax=b$, then an x which minimizes the mismatch between Ax and b is computed.

In the case where many x satisfy one of the criterion above, then a smallest (in terms of vector norm) such x is computed.

So, mismatch is handled first. Among all equally suitable x vectors that minimize the mismatch, choose a smallest one.