Least Squares

Geometric Algorithms
Lecture 23

CAS CS 132

Recap Problem

$$\mathbf{u} = \begin{bmatrix} 1 \\ 3 \\ -2 \\ -1 \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

Find the orthogonal projection of $\mathbf{u}$ *onto the span of* $\mathbf{v}$

Answer

$$\hat{\mathbf{u}} = \begin{bmatrix} 0 \\ 5/2 \\ -5/2 \\ 0 \end{bmatrix}$$

Objectives

1. Introduce the least squares problem as a method of *approximating* solutions to matrix equations
2. Learn how to solve the least squares problems
3. Connect least squares solutions to projections

Keywords

general least squares problem

sum of squares error (ℓ_2–error)

least squares solutions

orthogonal projections

normal equations

Orthogonal Matrices

Orthonormal Matrices

Definition. A matrix is **orthonormal** if its columns form an orthonormal set

The notes call a square orthonormal matrix an **orthogonal** matrix

Orthonormal Matrices

Definition. A matrix is **orthonormal** if its columns form an orthonormal set

The notes call a square orthonormal matrix an **orthogonal** matrix

This is incredibly confusing, but we'll try to be consistent and clear

Inverses of Orthogonal Matrices

Theorem. If an $n \times n$ matrix U is orthogonal (square orthonormal) then it is invertible and

$$U^{-1} = U^T$$

Orthonormal Matrices and Inner Products

Theorem. For a $m \times n$ orthonormal matrix U, and any vectors x and y in R^n

$$\langle Ux, Uy \rangle = \langle x, y \rangle$$

Orthonormal matrices preserve inner products

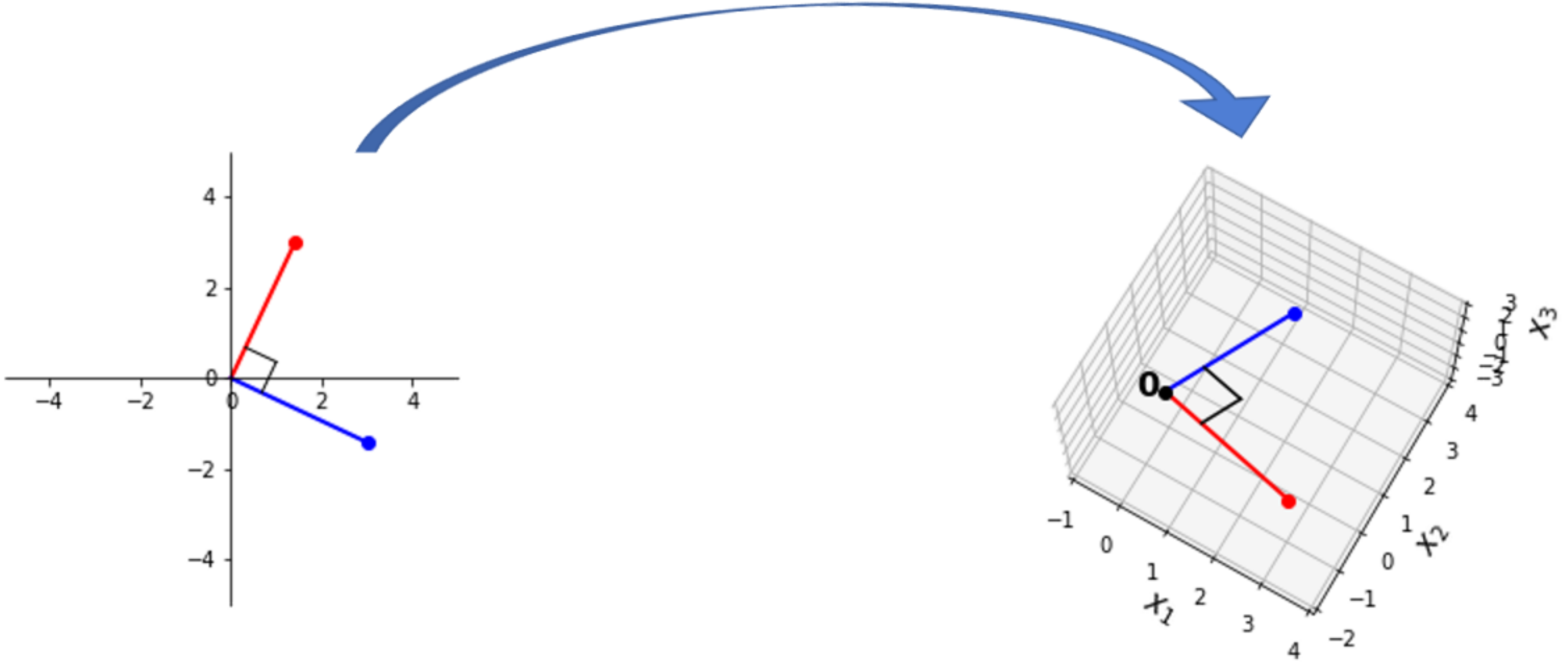
Verify:

Length, Angle, Orthogonality Preservation

Since lengths and angles are defined in terms of inner products, they are also preserved by orthonormal matrices:

The Picture

Orthonormal U



Example

$$U = \begin{bmatrix} 1/\sqrt{2} & 2/3 \\ 1/\sqrt{2} & -2/3 \\ 0 & 1/3 \end{bmatrix} \qquad x = \begin{bmatrix} \sqrt{2} \\ 3 \end{bmatrix}$$

moving on...

Motivation

The story of an enterprising CS132 student

The story of an enterprising CS132 student

Problem. Solve the equation $A\mathbf{x} = \mathbf{b}$

The story of an enterprising CS132 student

Problem. Solve the equation $A\mathbf{x} = \mathbf{b}$

Answer. Use `np.linalg.solve(A, b)`

The story of an enterprising CS132 student

Problem. Solve the equation $A\mathbf{x} = \mathbf{b}$

Answer. Use np.linalg.solve(A, b)

```
>>> A = np.array([
...     [1., 0, 5],
...     [1, -1, 4],
...     [0, 2, 2]])
>>> b = np.array([-1, 2, 3])
>>> np.linalg.solve(A, b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/homebrew/lib/python3.11/site-packages/numpy/linalg/linalg.py", line 409, in solve
    r = gufunc(a, b, signature=signature, extobj=extobj)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/opt/homebrew/lib/python3.11/site-packages/numpy/linalg/linalg.py", line 112, in _raise_linalgerror_singular
    raise LinAlgError("Singular matrix")
numpy.linalg.LinAlgError: Singular matrix
```

The story of an enterprising CS132 student

Problem. Solve the equation $A\mathbf{x} = \mathbf{b}$

Answer. Use np.linalg.solve(A, b)

```
>>> A = np.array([
...     [1., 0, 5],
...     [1, -1, 4],
...     [0, 2, 2]])
>>> b = np.array([-1, 2, 3])
>>> np.linalg.solve(A, b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/homebrew/lib/python3.11/site-packages/numpy/linalg/linalg.py", line 409, in solve
    r = gufunc(a, b, signature=signature, extobj=extobj)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/homebrew/lib/python3.11/site-packages/numpy/linalg/linalg.py", line 112, in _raise_linalgerror_singular
    raise LinAlgError("Singular matrix")
numpy.linalg.LinAlgError: Singular matrix
```

This doesn't always work

Reads the docs...

numpy.linalg.solve

`linalg.solve(a, b)` [source]

Solve a linear matrix equation, or system of linear scalar equations.

Computes the “exact” solution, *x*, of the well-determined, i.e., full rank, linear matrix equation *ax* = *b*.

Parameters:

a : *(..., M, M) array_like*

Coefficient matrix.

b : *{(..., M,), (..., M, K)}, array_like*

Ordinate or “dependent variable” values.

Returns:

x : *{(..., M,), (..., M, K)} ndarray*

Solution to the system a x = b. Returned shape is identical to *b*.

Raises:

LinAlgError

If *a* is singular or not square.

See also

`scipy.linalg.solve`

Reads the docs...

numpy.linalg.solve

`linalg.solve(a, b)` [source]

Solve a linear matrix equation, or system of linear scalar equations.

Computes the "exact" solution, *x*, of the well-determined, i.e., full rank, linear matrix equation *ax* = *b*.

Parameters: **a** : *(..., M, M) array_like*

Coefficient matrix.

b : *{(..., M,), (..., M, K)}, array_like*

Ordinate or "dependent variable" values.

Returns: **x** : *{(..., M,), (..., M, K)} ndarray*

Solution to the system a x = b. Returned shape is identical to *b*.

Raises: LinAlgError

If *a* is singular or not square.

See also

`scipy.linalg.solve`

Similar function in SciPy.

Reads the docs…

b : {(…, M,), (…, M, K)}, array_like

Ordinate or "dependent variable" values.

Returns: **x** : *{(…, M,), (…, M, K)} ndarray*

Solution to the system a x = b. Returned shape is identical to *b*.

Raises: LinAlgError

If *a* is singular or not square.

See also

`scipy.linalg.solve`

Similar function in SciPy.

Notes

New in version 1.8.0.

Broadcasting rules apply, see the `numpy.linalg` documentation for details.

The solutions are computed using LAPACK routine `_gesv`.

a must be square and of full-rank, i.e., all rows (or, equivalently, columns) must be linearly independent; if either is not true, use `lstsq` for the least-squares best "solution" of the system/equation.

Reads the docs...

b : {(..., M,), (..., M, K)}, array_like

Ordinate or "dependent variable" values.

Returns: **x** : *{(..., M,), (..., M, K)} ndarray*

Solution to the system a x = b. Returned shape is identical to *b*.

Raises: LinAlgError

If *a* is singular or not square.

See also

`scipy.linalg.solve`

Similar function in SciPy.

Notes

New in version 1.8.0.

Broadcasting rules apply, see the `numpy.linalg` documentation for details.

The solutions are computed using LAPACK routine `_gesv`.

a must be square and of full-rank, i.e., all rows (or, equivalently, columns) must be linearly independent; if either is not true, use `lstsq` for the least-squares best "solution" of the system/equation.

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

uh...probably numerical errors...

Answer: $\mathbf{x} = \begin{bmatrix} -1/9 \\ 7/9 \\ 2/9 \end{bmatrix}$

np.linalg.lstsq

```
>>> np.linalg.lstsq(A, b)
<stdin>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)``
where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old,
explicitly pass `rcond=-1`.
(array([-0.11111111,  0.77777778,  0.22222222]), array([], dtype=float64), 2, array([6.84168488e+00,
2.27845297e+00, 6.13801942e-17]))
>>> x = np.array([-0.11111111,  0.77777778,  0.22222222])
>>> A @ x
array([ 9.99999990e-01, -9.99999994e-09,  2.00000000e+00])
>>>
```

uh...probably numerical errors...

Answer: $\mathbf{x} = \begin{bmatrix} -1/9 \\ 7/9 \\ 2/9 \end{bmatrix}$ **This is not correct**

This System is Inconsistent

$$\begin{bmatrix} 1 & 0 & 5 & -1 \\ 1 & -1 & 4 & 2 \\ 0 & 2 & 2 & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 5 & -1 \\ 0 & -1 & -1 & 3 \\ 0 & 2 & 2 & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 5 & -1 \\ 0 & -1 & -1 & 3 \\ 0 & 0 & 0 & 9 \end{bmatrix}$$

The "correct" answer: There is no solution

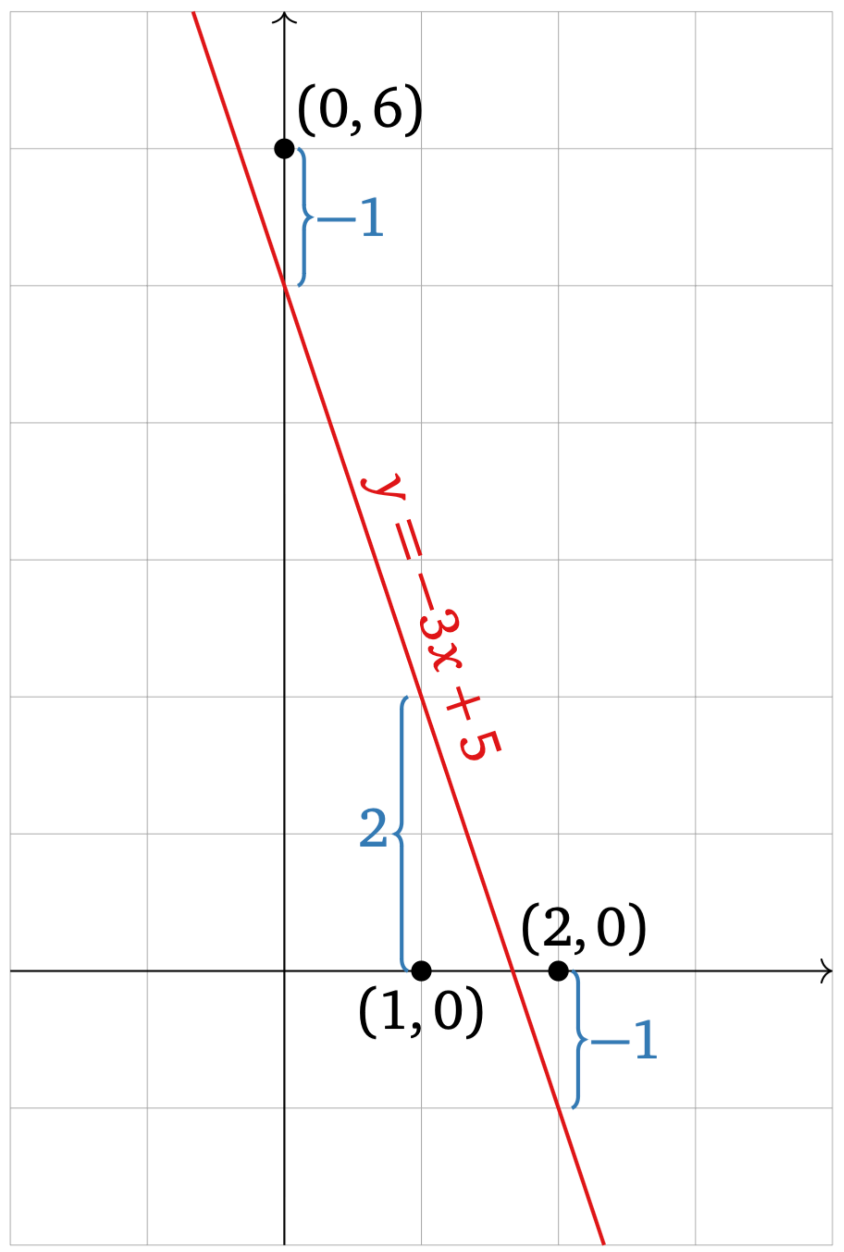
This System is Inconsistent

$$\begin{bmatrix} 1 & 0 & 5 & -1 \\ 1 & -1 & 4 & 2 \\ 0 & 2 & 2 & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 5 & -1 \\ 0 & -1 & -1 & 3 \\ 0 & 2 & 2 & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 5 & -1 \\ 0 & -1 & -1 & 3 \\ 0 & 0 & 0 & 9 \end{bmatrix}$$

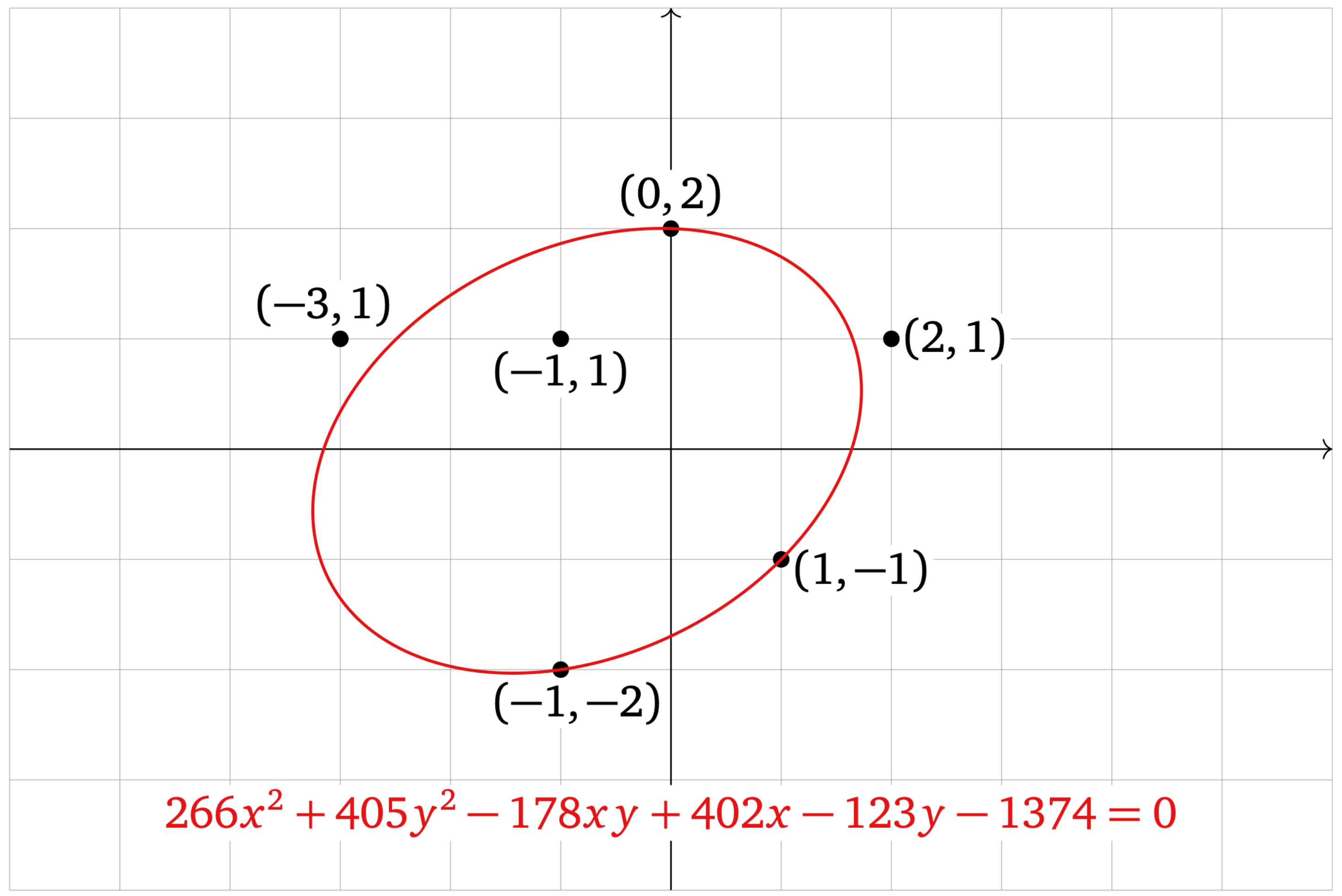
The "correct" answer: There is no solution

What's going on here?

Non-Linearity



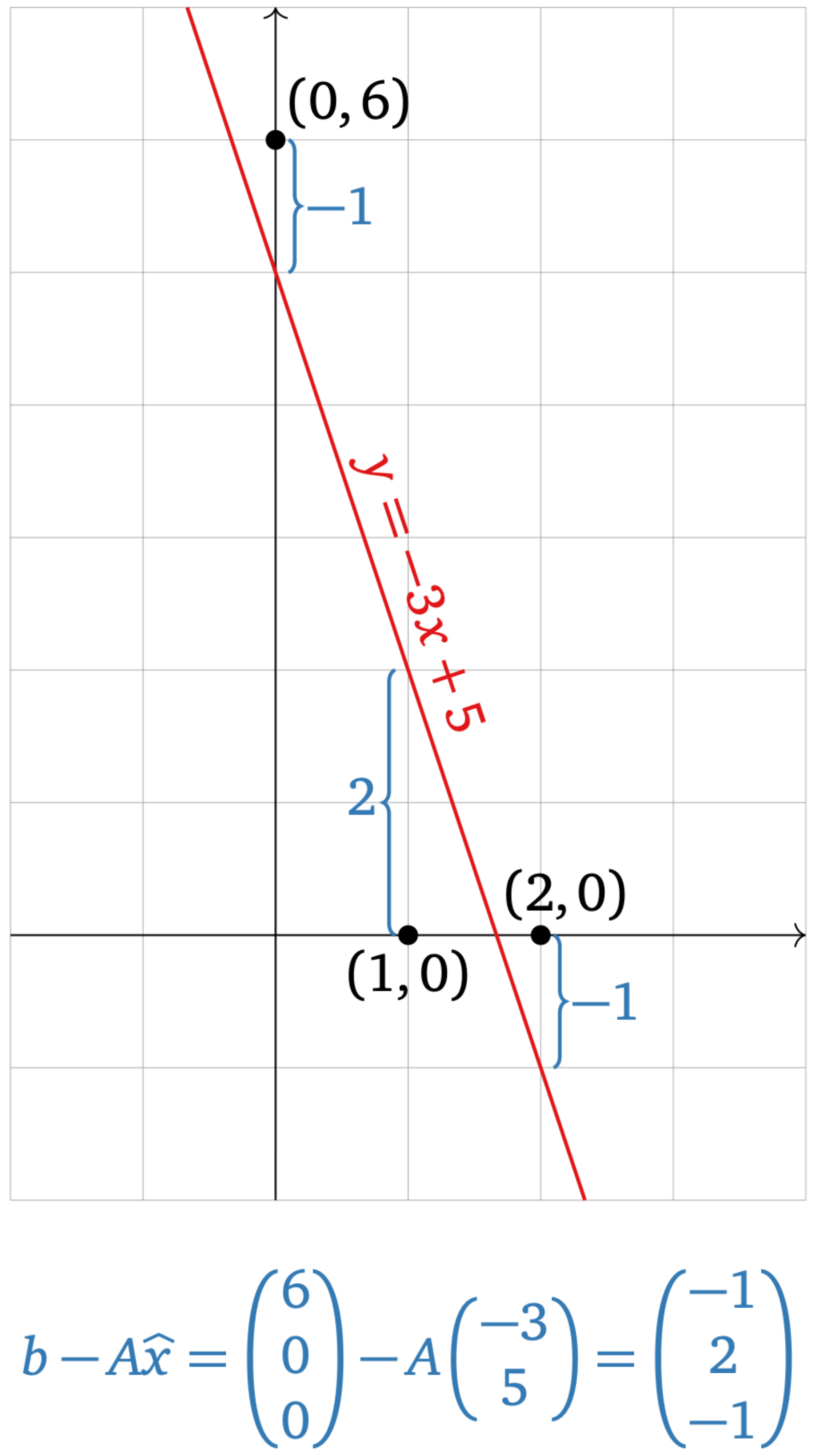
$$b - A\hat{x} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} - A\begin{pmatrix} -3 \\ 5 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}$$

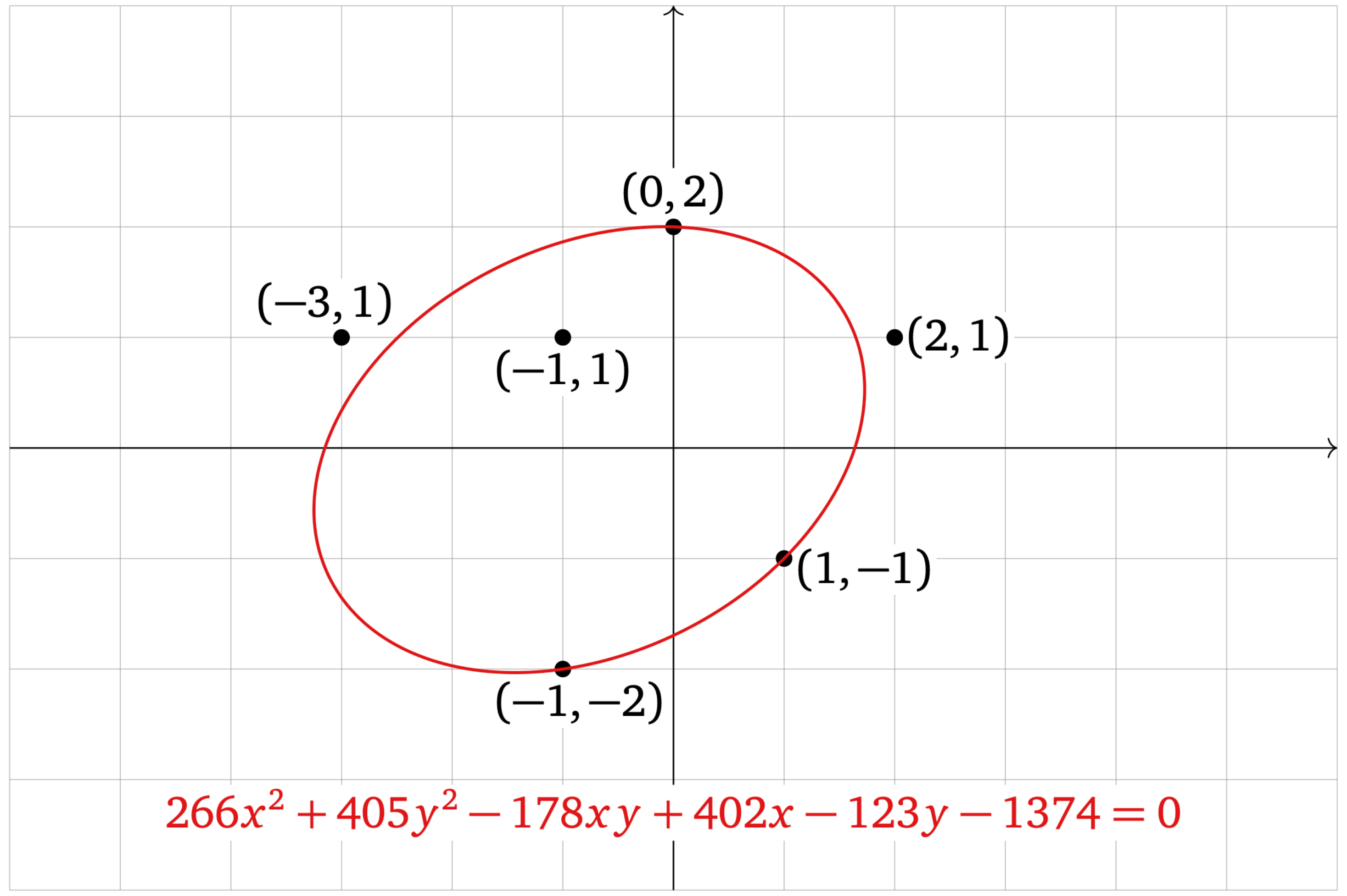


https://textbooks.math.gatech.edu/ila/least-squares.html

Non-Linearity

Linear algebra is very powerful and very clean, but **the world isn't linear.** There are non-linear relationships and sources of *noise*



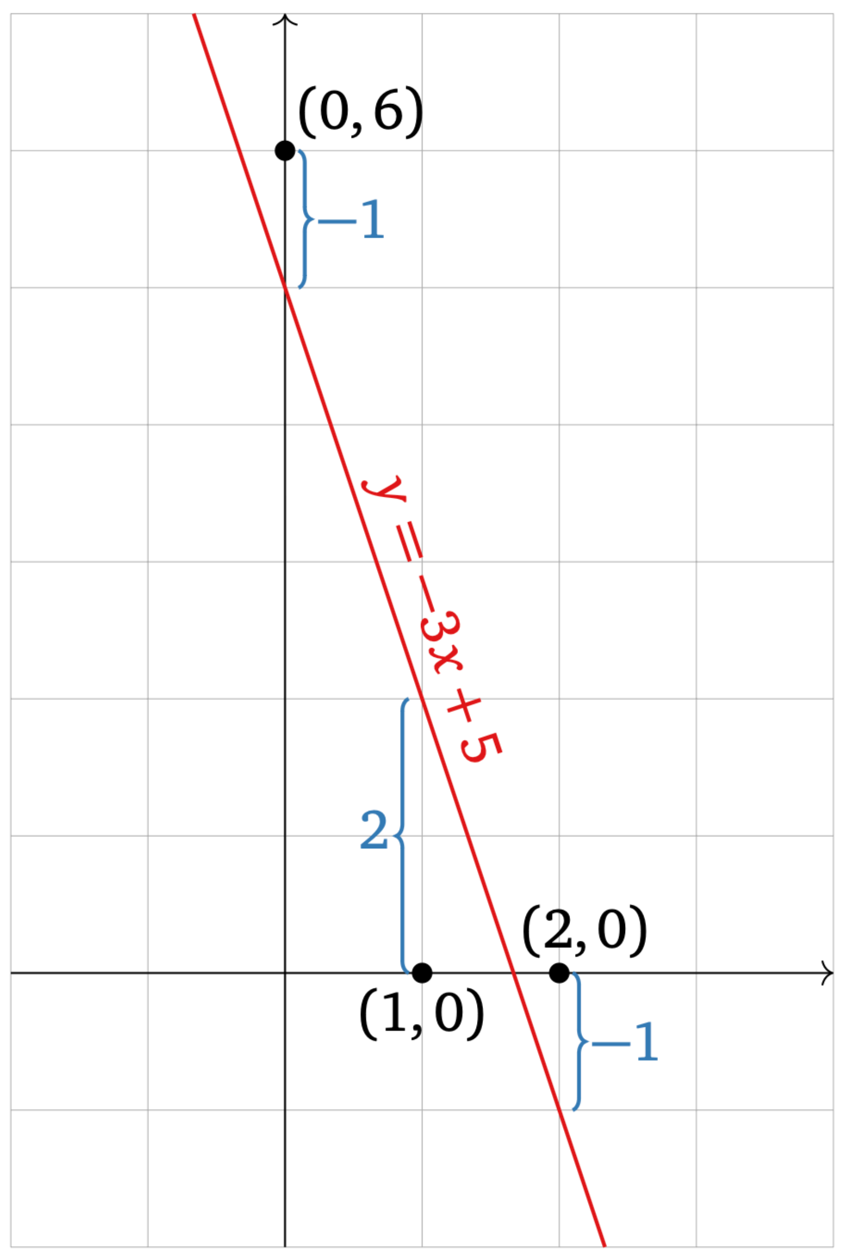


https://textbooks.math.gatech.edu/ila/least-squares.html

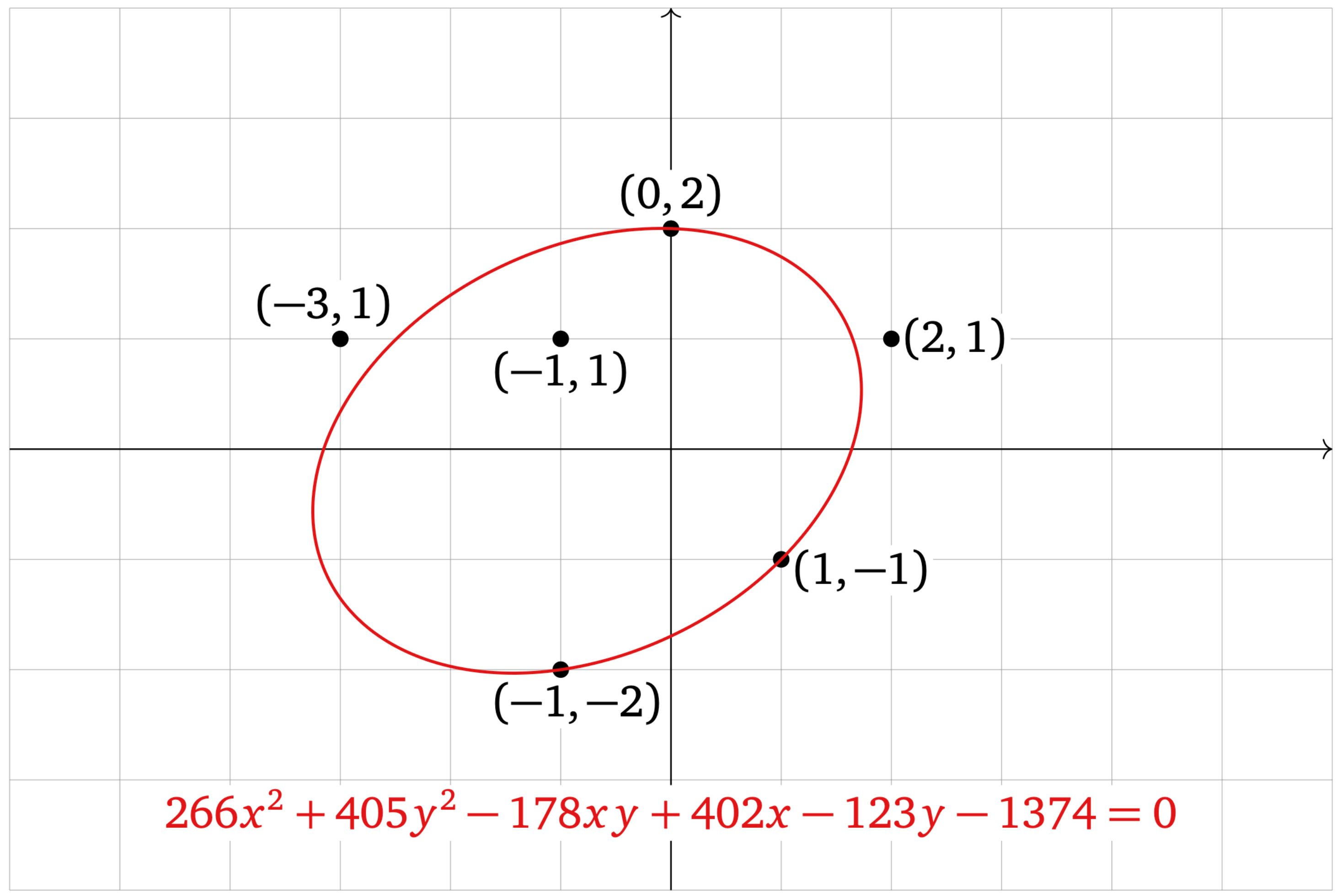
Non-Linearity

Linear algebra is very powerful and very clean, but **the world isn't linear.** There are non-linear relationships and sources of *noise*

We can't force the world to be linear



$$b - A\hat{x} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} - A\begin{pmatrix} -3 \\ 5 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}$$



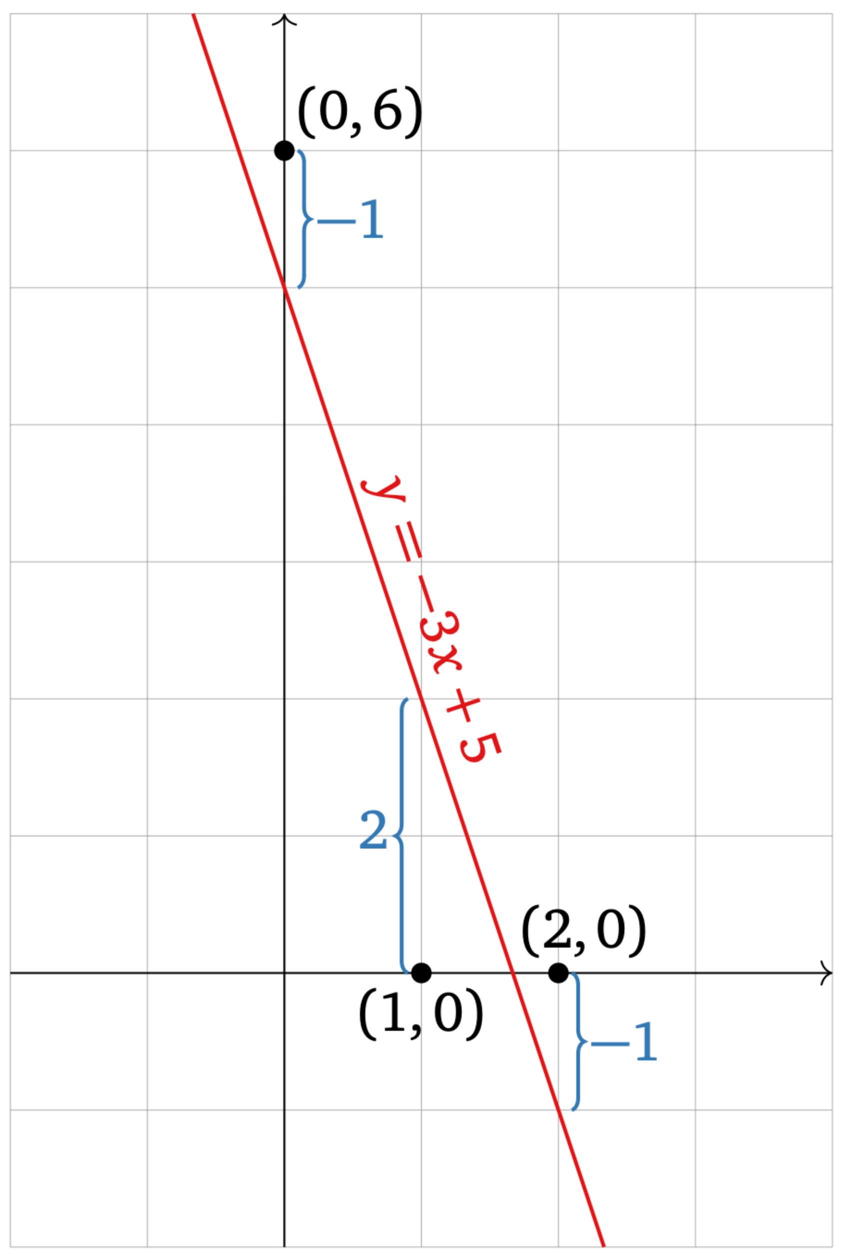
https://textbooks.math.gatech.edu/ila/least-squares.html

Non-Linearity

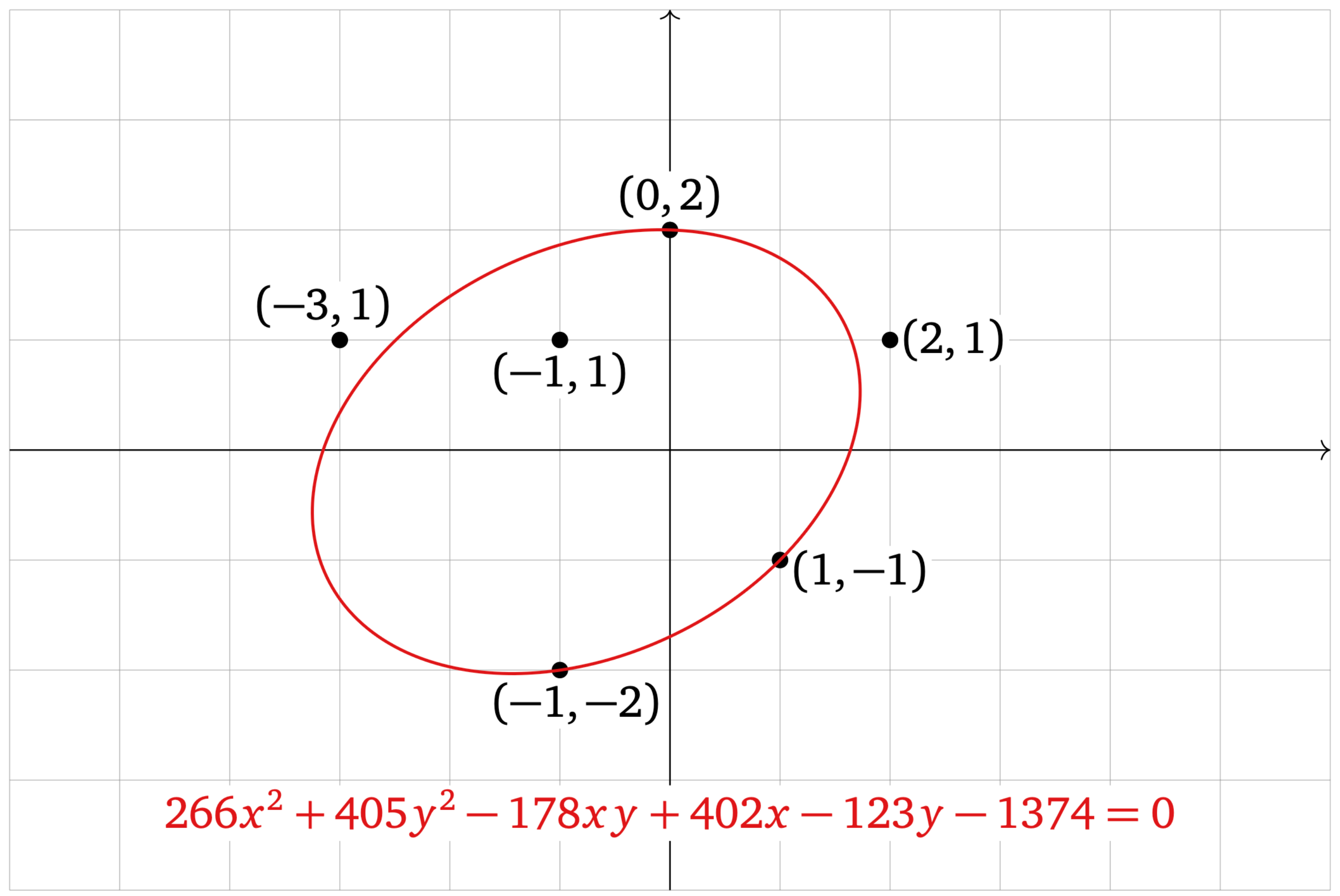
Linear algebra is very powerful and very clean, but **the world isn't linear.** There are non-linear relationships and sources of *noise*

We can't force the world to be linear

But we can try...

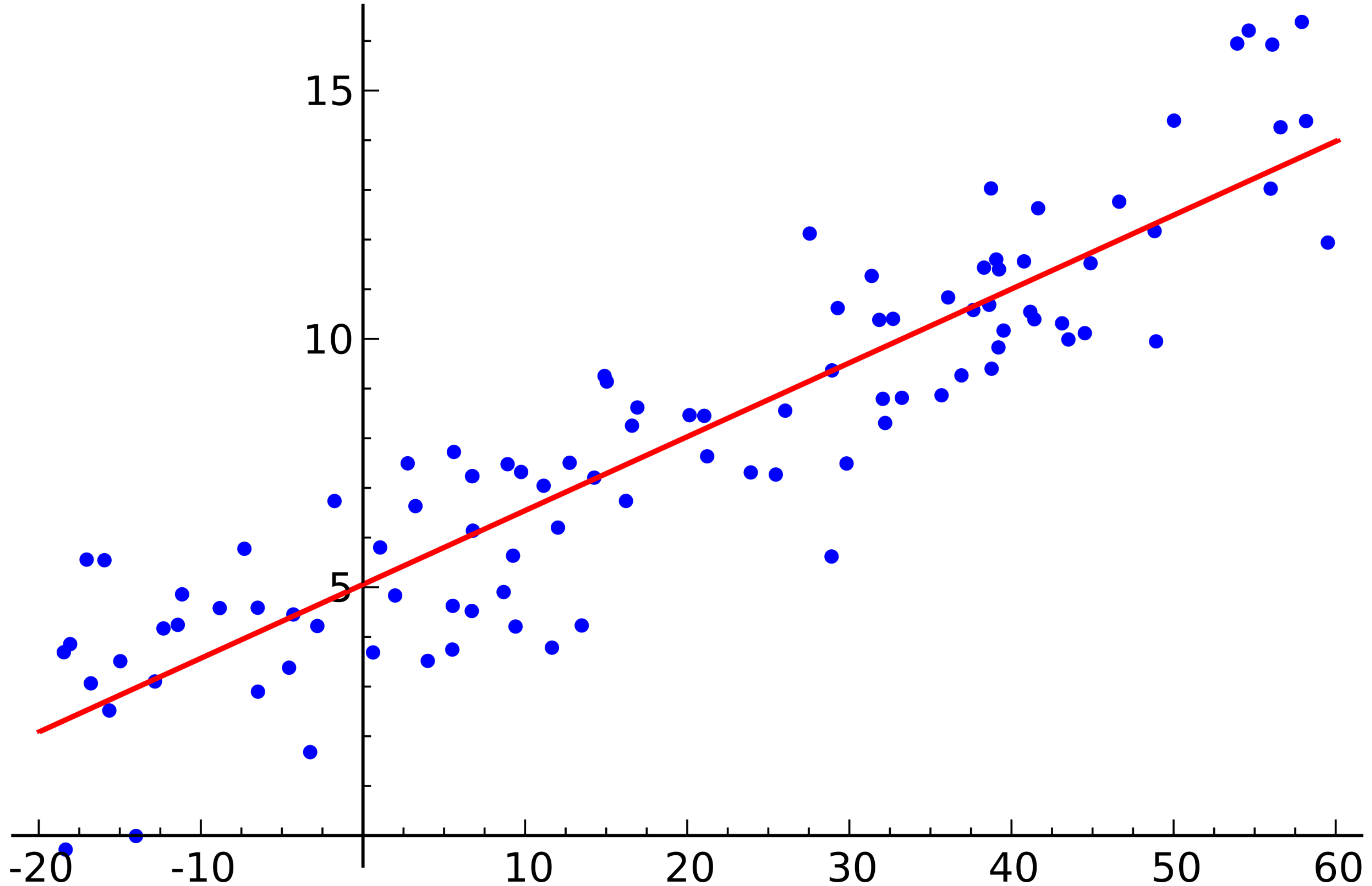


$$b - A\hat{x} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} - A\begin{pmatrix} -3 \\ 5 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}$$



https://textbooks.math.gatech.edu/ila/least-squares.html

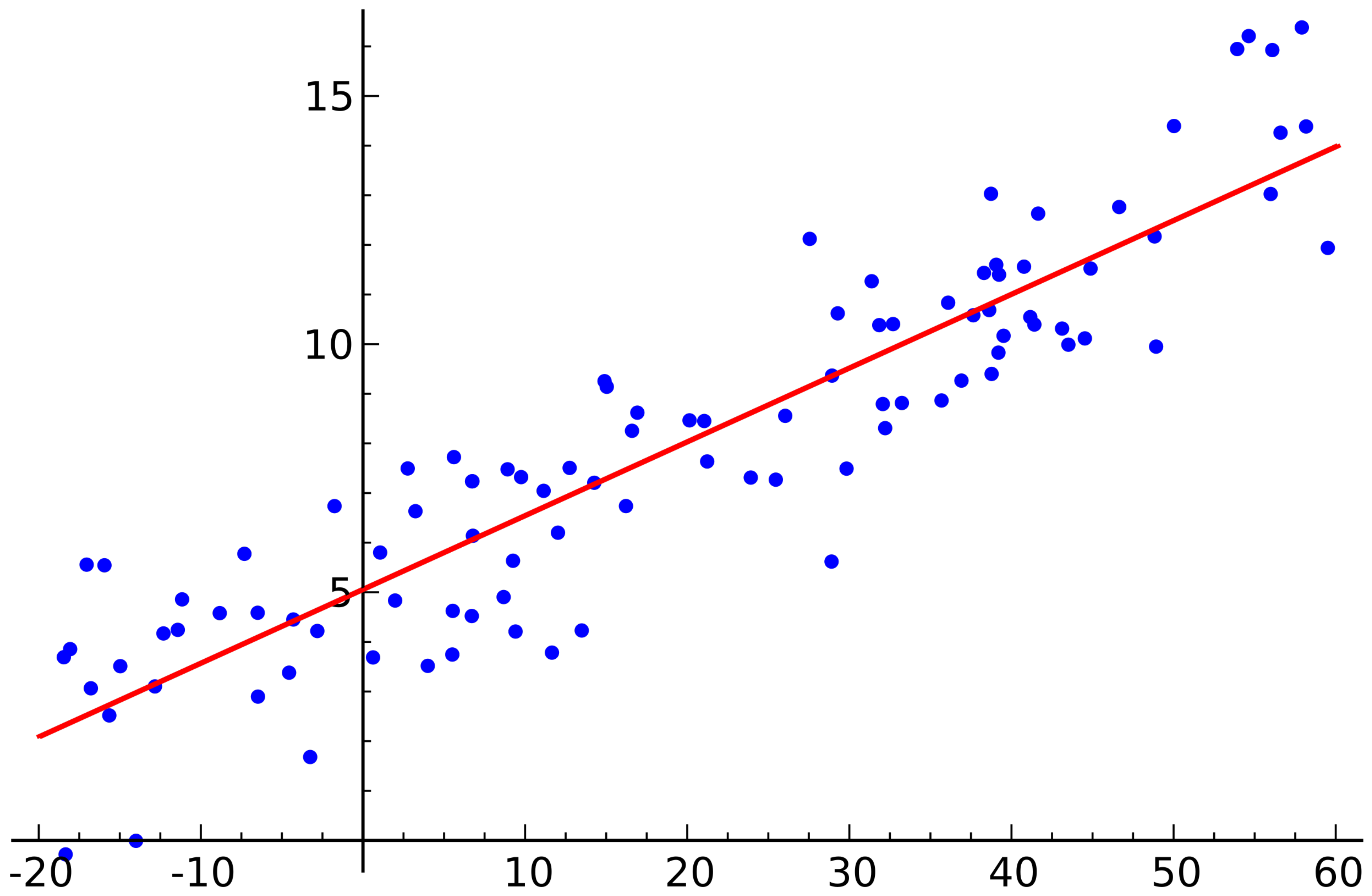
The Idea



https://commons.wikimedia.org/wiki/File:Linear_regression.svg

The Idea

Least Squares is a method for finding *approximate* solutions to systems of linear equations

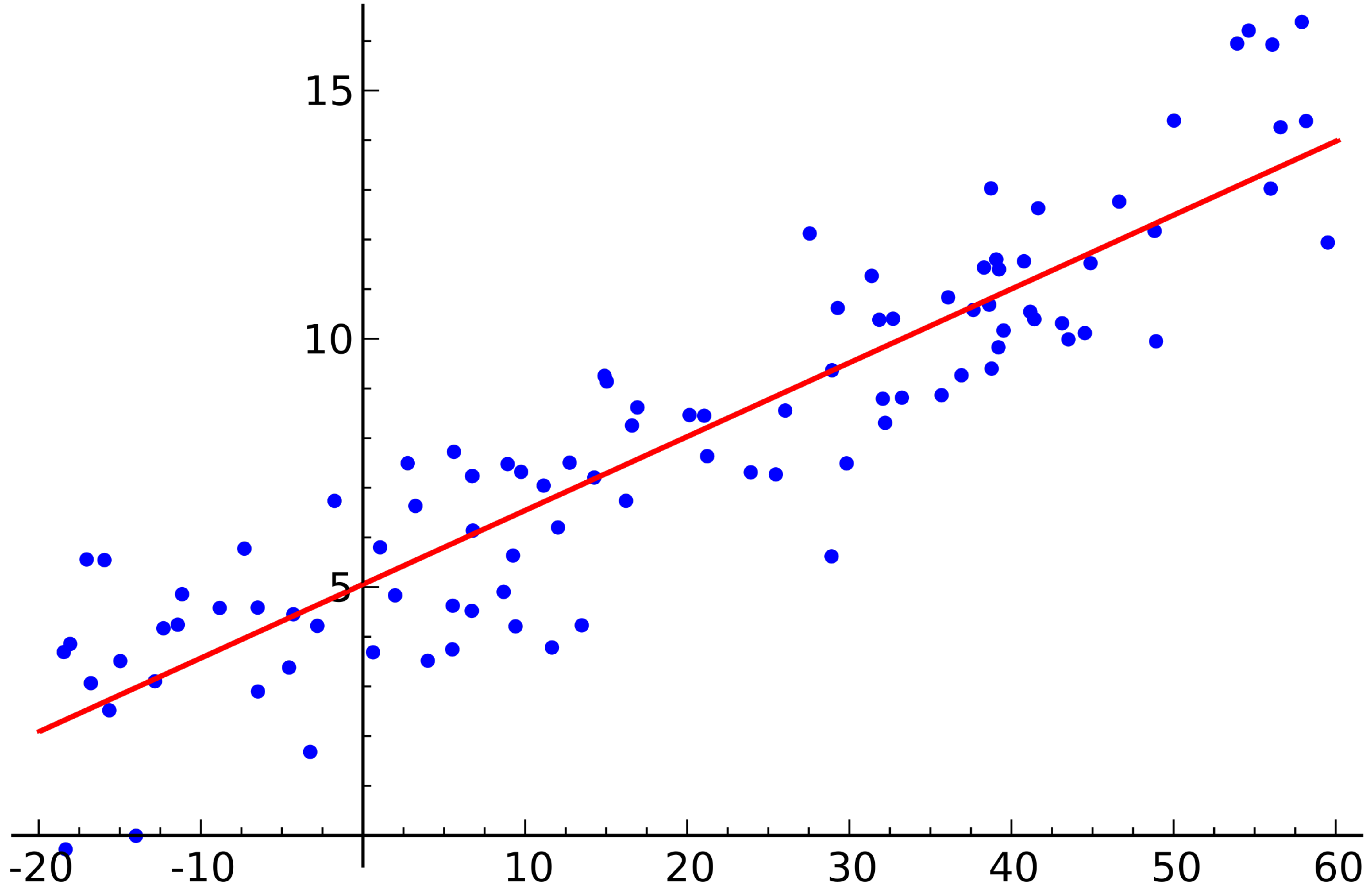


https://commons.wikimedia.org/wiki/File:Linear_regression.svg

The Idea

Least Squares is a method for finding *approximate* solutions to systems of linear equations

This is **a lot more useful in practice** than exact solutions



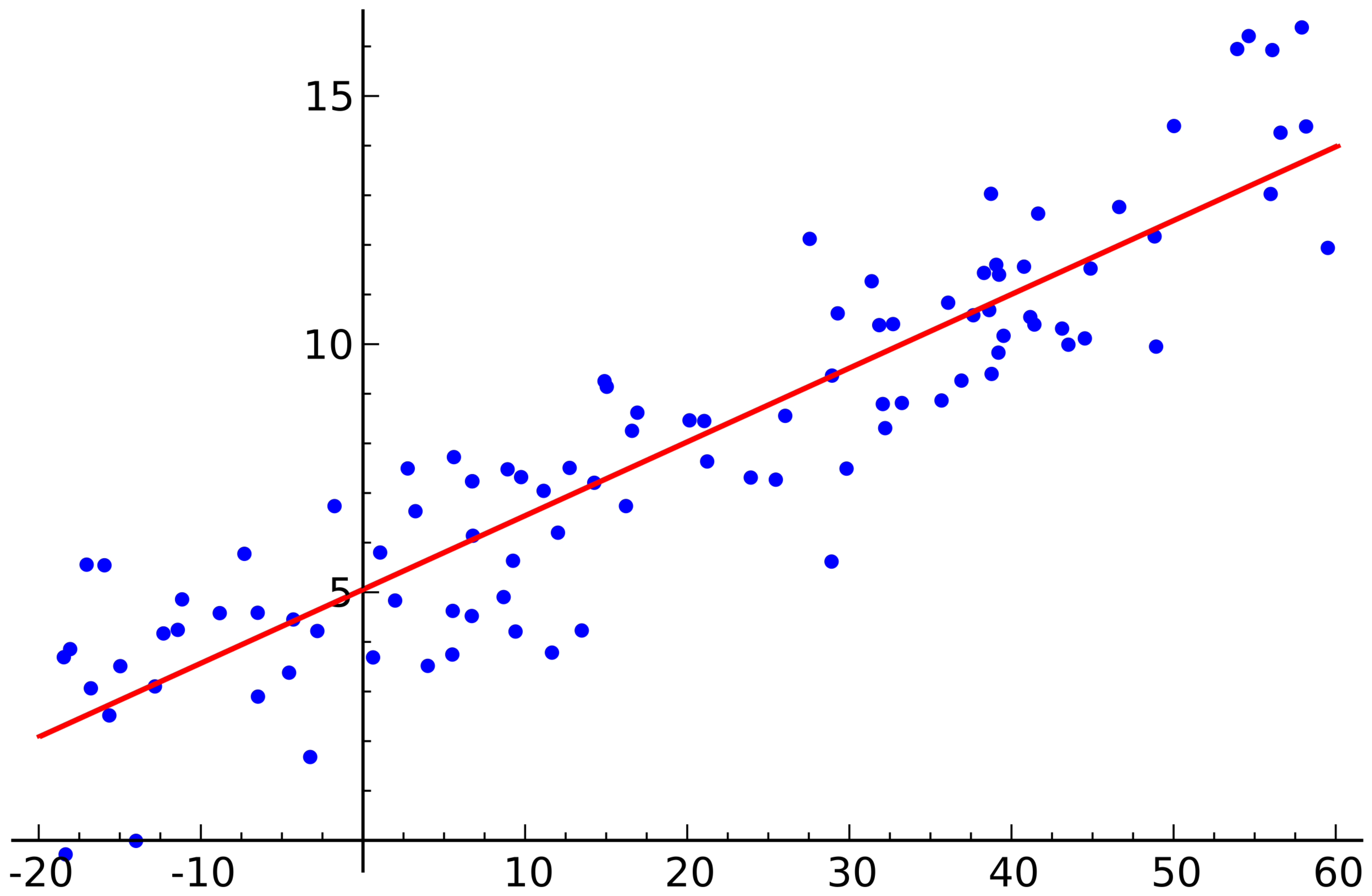
https://commons.wikimedia.org/wiki/File:Linear_regression.svg

The Idea

Least Squares is a method for finding *approximate* solutions to systems of linear equations

This is **a lot more useful in practice** than exact solutions

It can be used to do **linear regression** from stats class



https://commons.wikimedia.org/wiki/File:Linear_regression.svg

General Least Squares Problem

The Picture

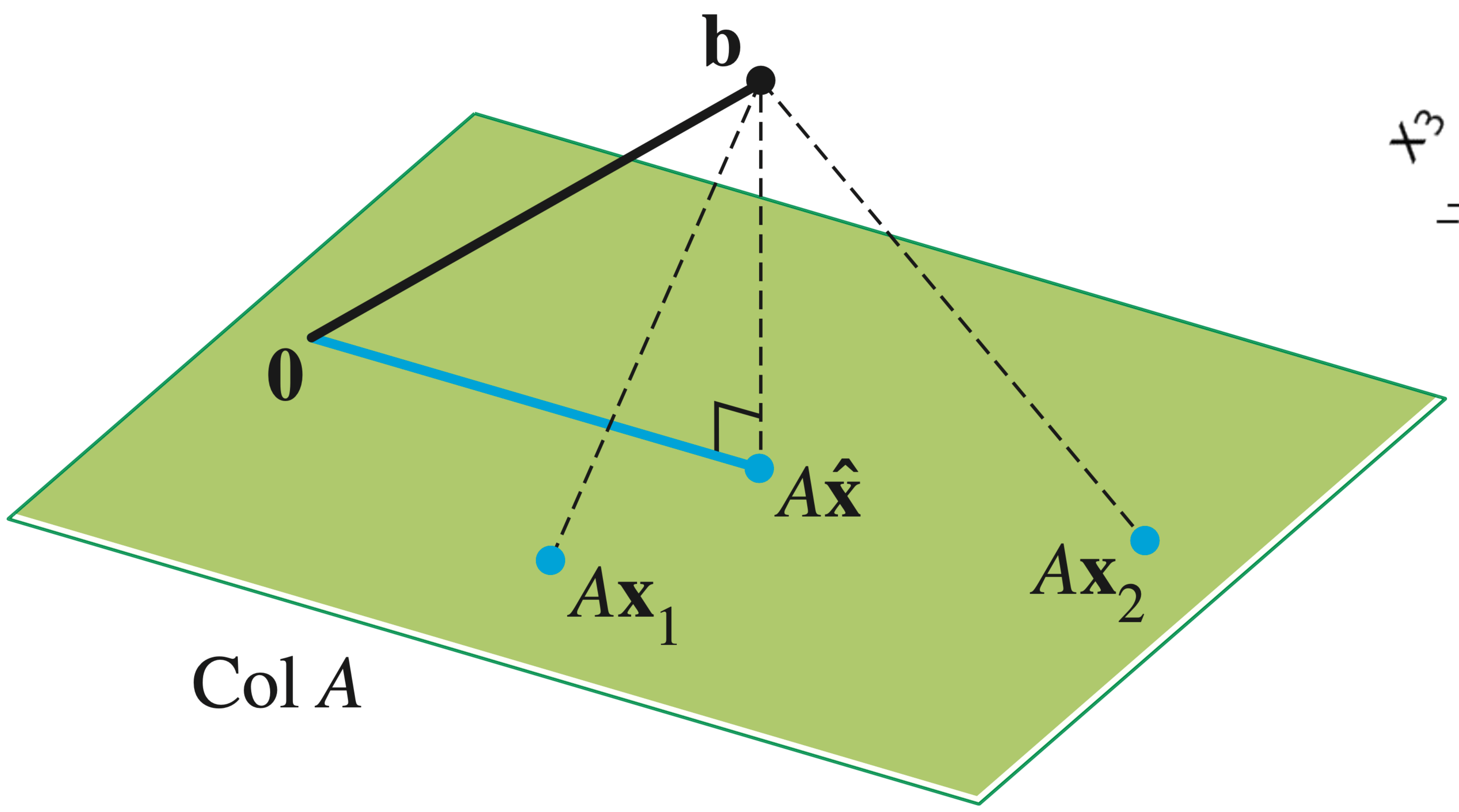
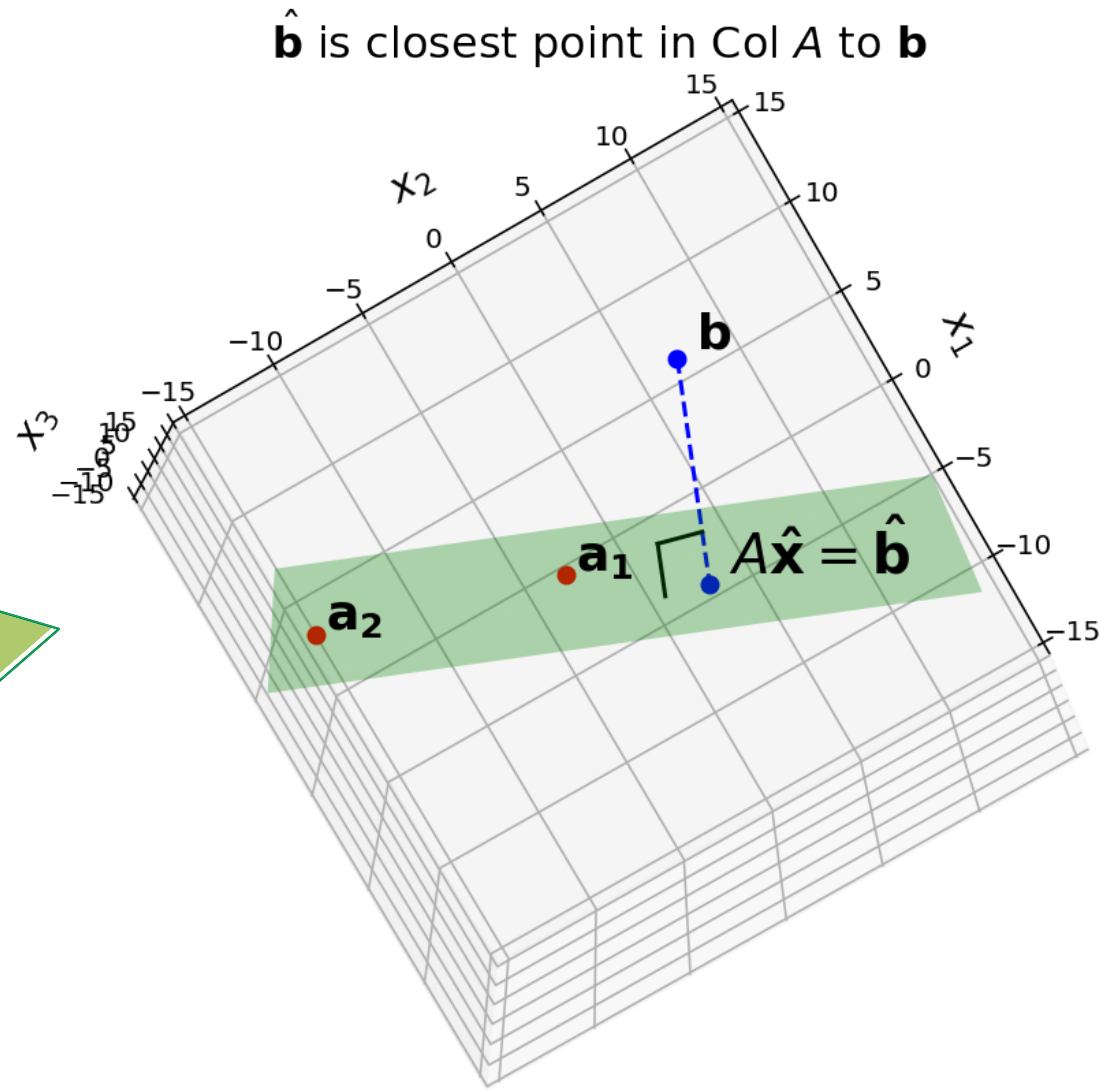
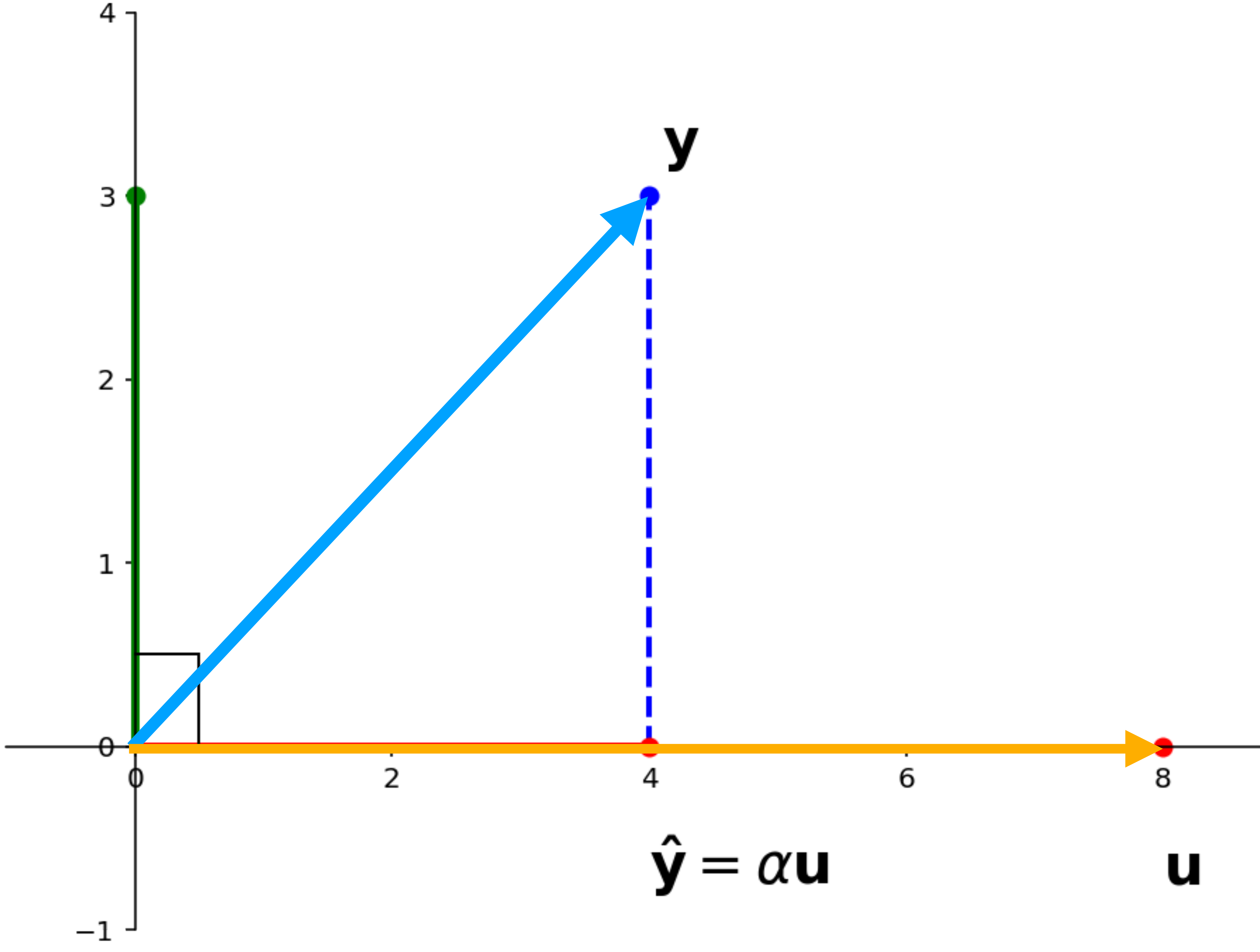


Figure 22.8

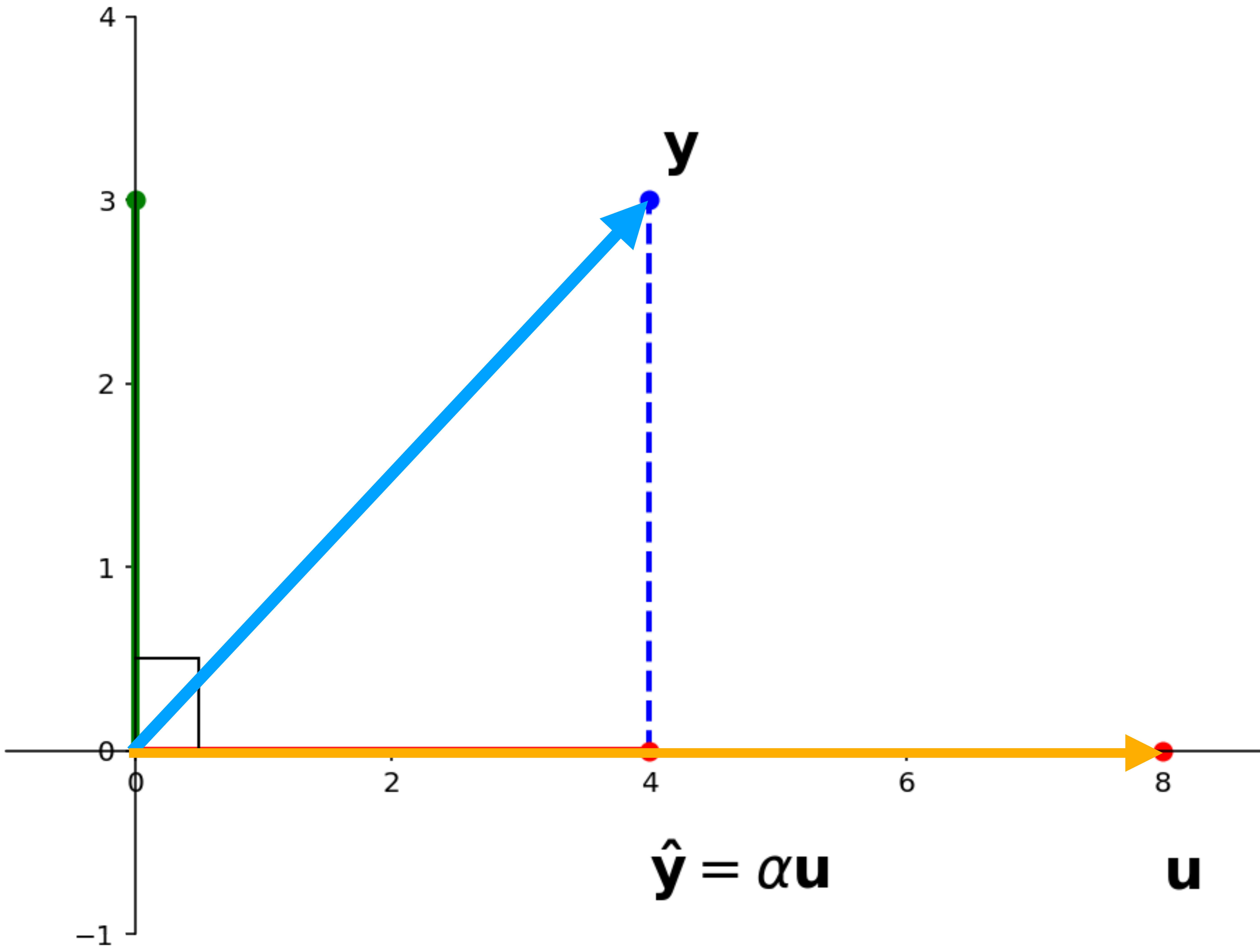


Recall: Orthogonal Projection



Recall: Orthogonal Projection

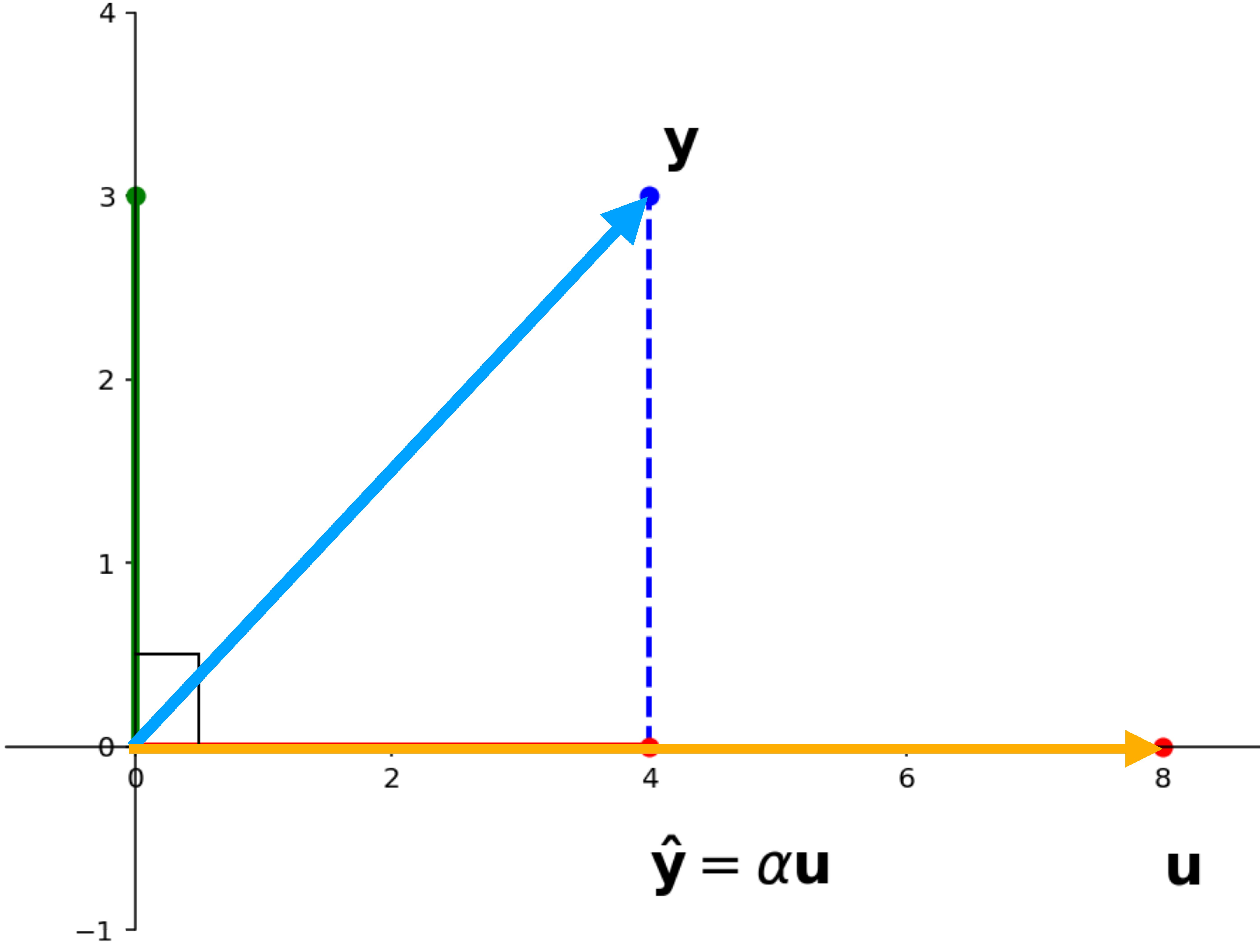
Question. Given vectors $\mathbf{y}$ and $\mathbf{u}$ in R^n, find vectors $\hat{\mathbf{y}}$ and $\mathbf{z}$ such that



Recall: Orthogonal Projection

Question. Given vectors $\mathbf{y}$ and $\mathbf{u}$ in R^n, find vectors $\hat{\mathbf{y}}$ and $\mathbf{z}$ such that

» $\mathbf{z}$ is orthogonal to $\mathbf{u}$ (i.e., $\mathbf{z} \cdot \mathbf{u} = 0$)

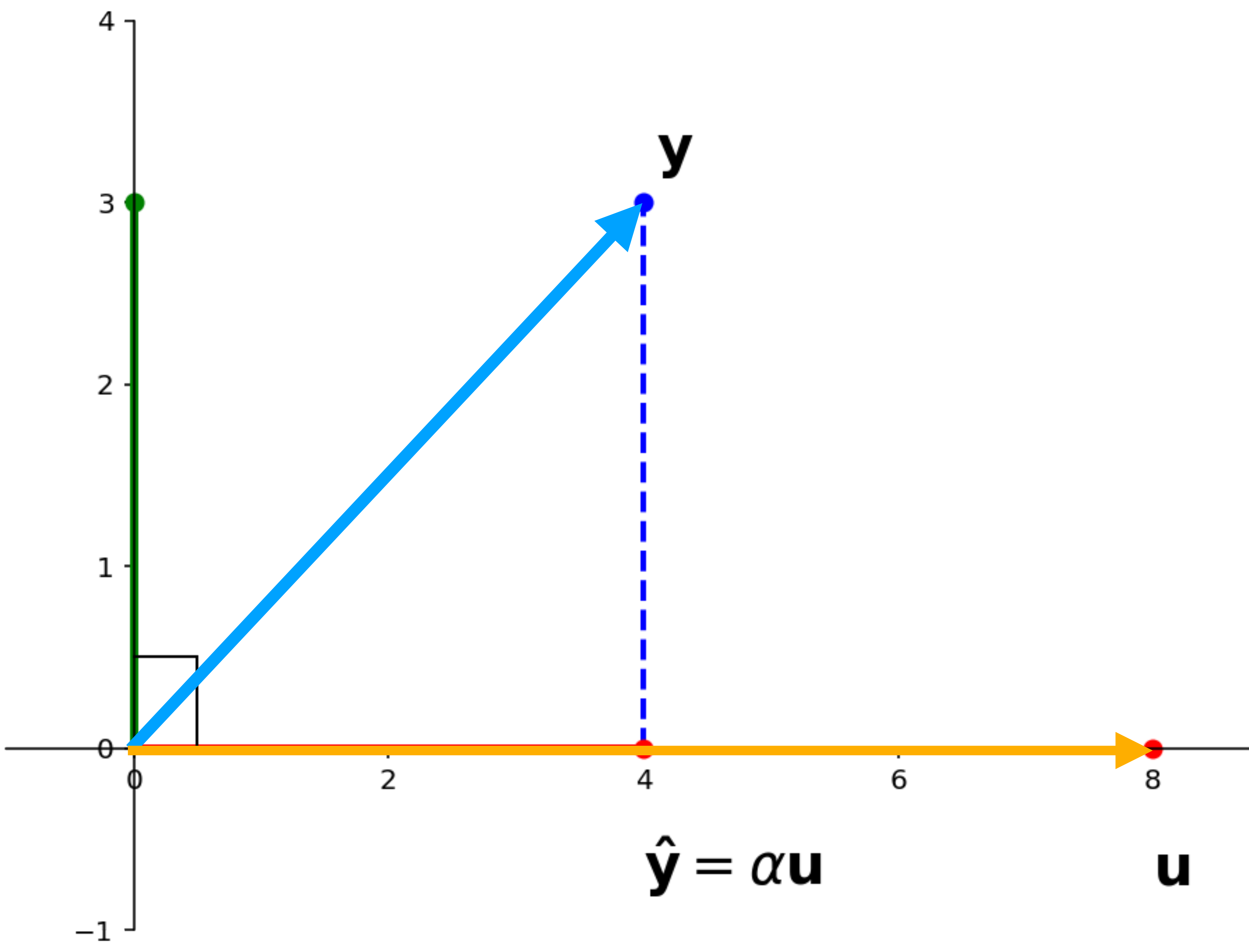


Recall: Orthogonal Projection

Question. Given vectors $\mathbf{y}$ and $\mathbf{u}$ in R^n, find vectors $\hat{\mathbf{y}}$ and $\mathbf{z}$ such that

» $\mathbf{z}$ is orthogonal to $\mathbf{u}$ (i.e., $\mathbf{z} \cdot \mathbf{u} = 0$)

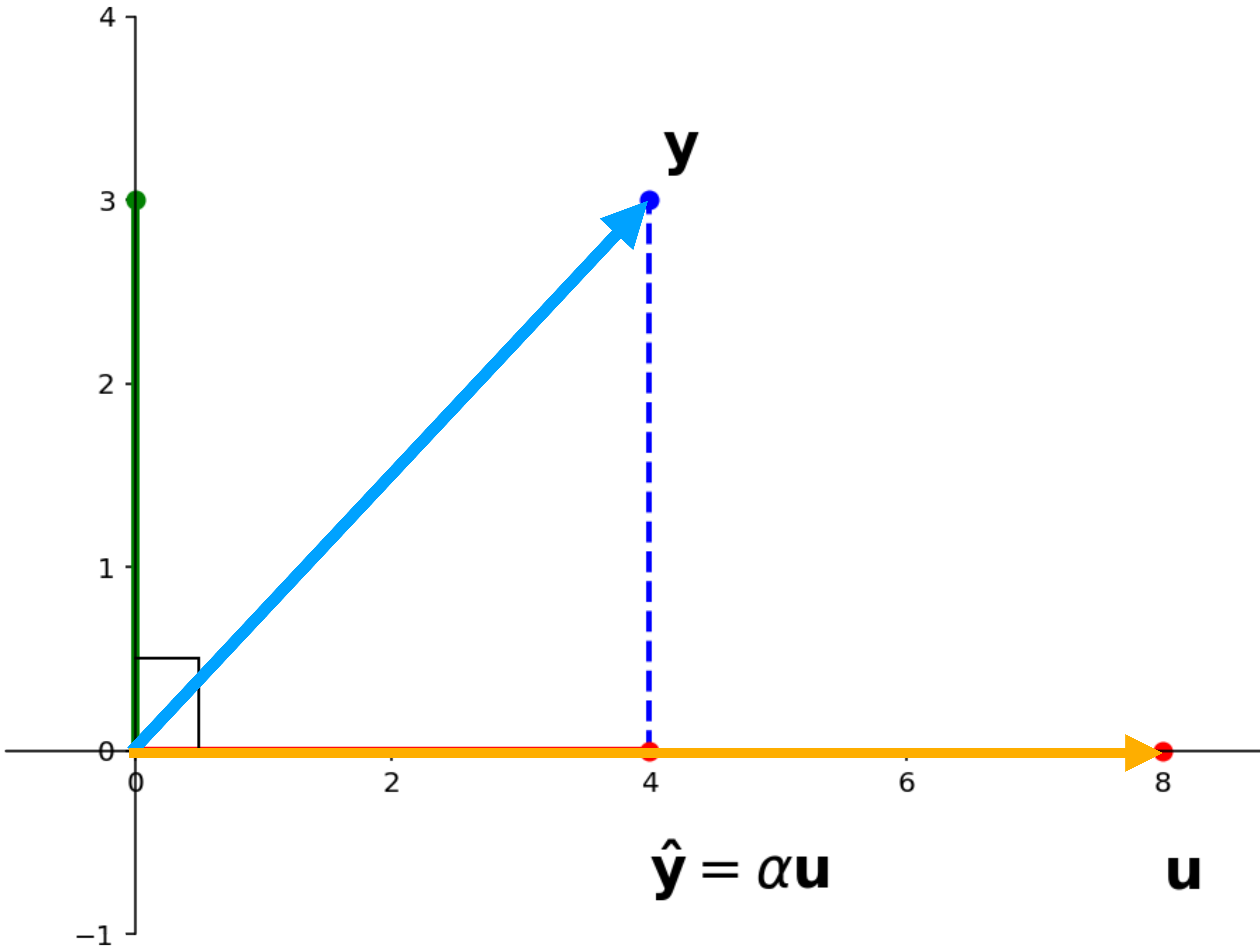
» $\hat{\mathbf{y}} \in span\{\mathbf{u}\}$



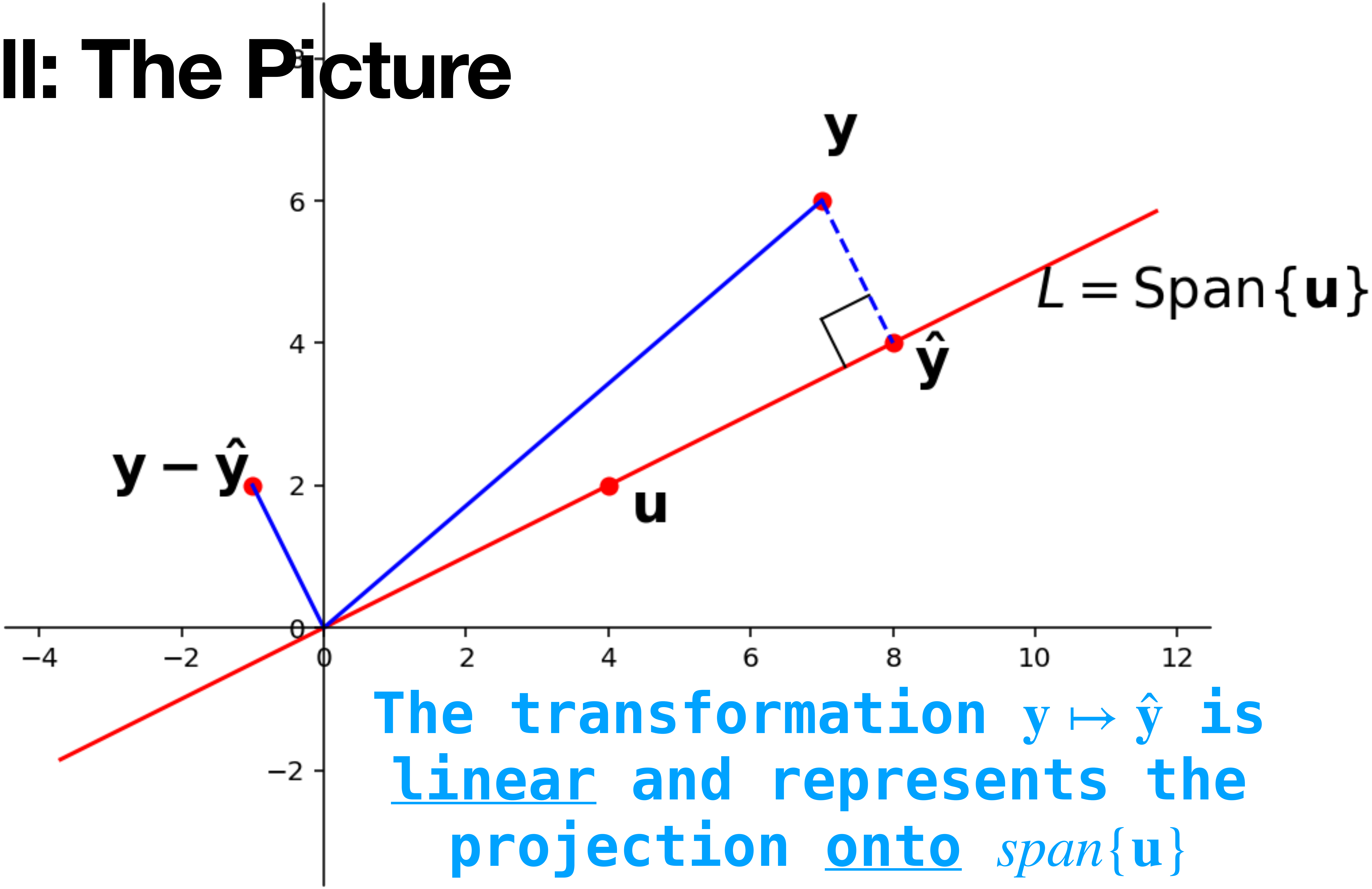
Recall: Orthogonal Projection

Question. Given vectors $\mathbf{y}$ and $\mathbf{u}$ in R^n, find vectors $\hat{\mathbf{y}}$ and $\mathbf{z}$ such that

- » $\mathbf{z}$ is orthogonal to $\mathbf{u}$ (i.e., $\mathbf{z} \cdot \mathbf{u} = 0$)
- » $\hat{\mathbf{y}} \in span\{\mathbf{u}\}$
- » $\mathbf{y} = \hat{\mathbf{y}} + \mathbf{z}$



Recall: The Picture



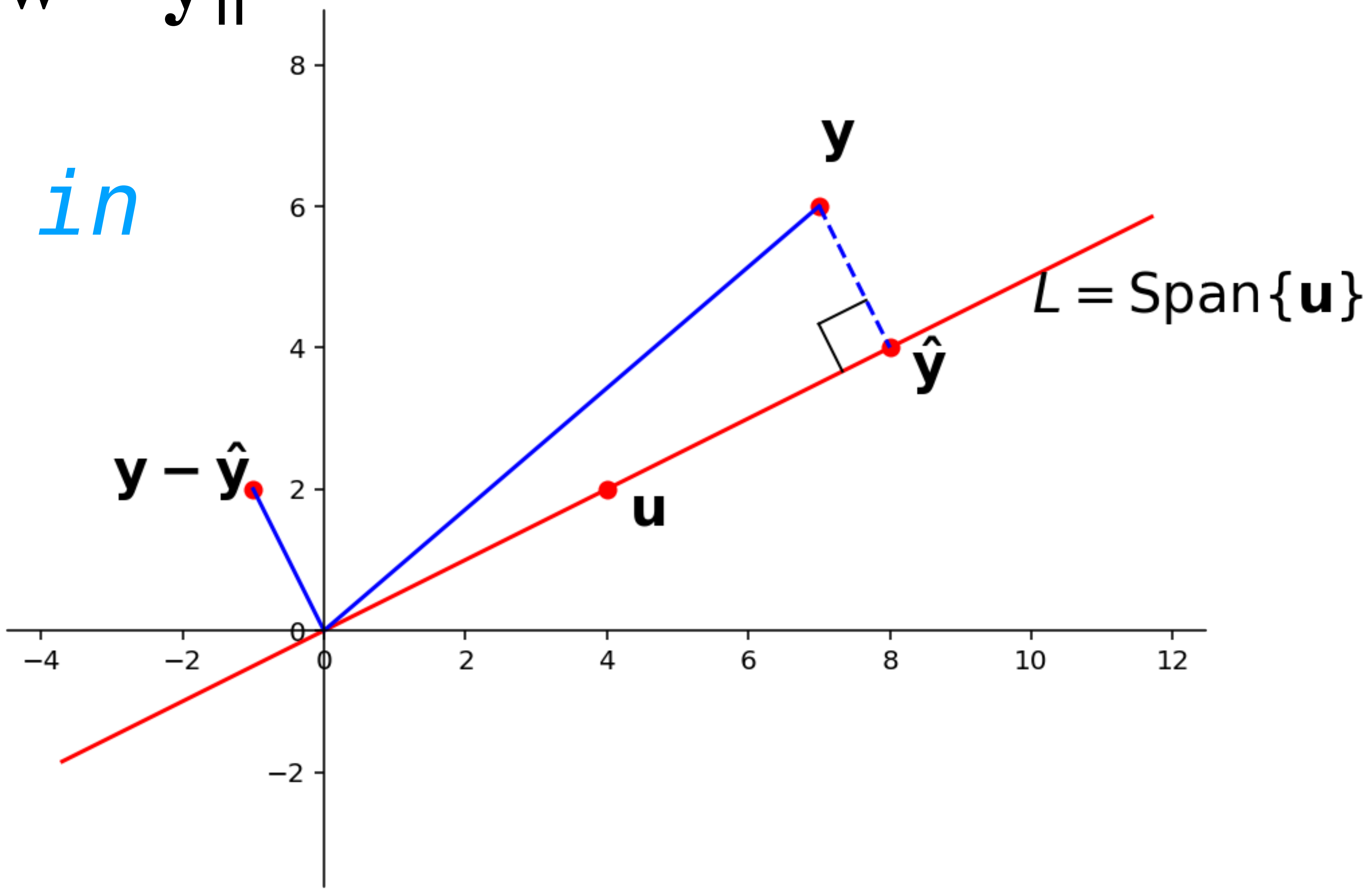
The transformation $\mathbf{y} \mapsto \hat{\mathbf{y}}$ is <u>linear</u> and represents the projection <u>onto</u> $span\{\mathbf{u}\}$

Recall: $\hat{\mathbf{y}}$ and Distance

Theorem. $\|\hat{\mathbf{y}} - \mathbf{y}\| = \min_{\mathbf{w} \in span\{\mathbf{u}\}} \|\mathbf{w} - \mathbf{y}\|$

$\hat{\mathbf{y}}$ is the closest vector in $span\{\mathbf{u}\}$ to $\mathbf{y}$

"Proof" by inspection:



The Equational Perspective

The Equational Perspective

We know the equation $x\mathbf{u} = \mathbf{y}$ may have no solution

The Equational Perspective

We know the equation $x\mathbf{u} = \mathbf{y}$ may have no solution

Question. Find a value α such that $\alpha\mathbf{u}$ is as close as possible to $\mathbf{y}$

The Equational Perspective

We know the equation $x\mathbf{u} = \mathbf{y}$ may have no solution

Question. Find a value α such that $\alpha\mathbf{u}$ is as close as possible to $\mathbf{y}$

That is, the distance $dist(\mathbf{y}, \alpha\mathbf{u}) = \|\mathbf{y} - \alpha\mathbf{u}\|$ is as small as possible

The Equational Perspective

We know the equation $x\mathbf{u} = \mathbf{y}$ may have no solution

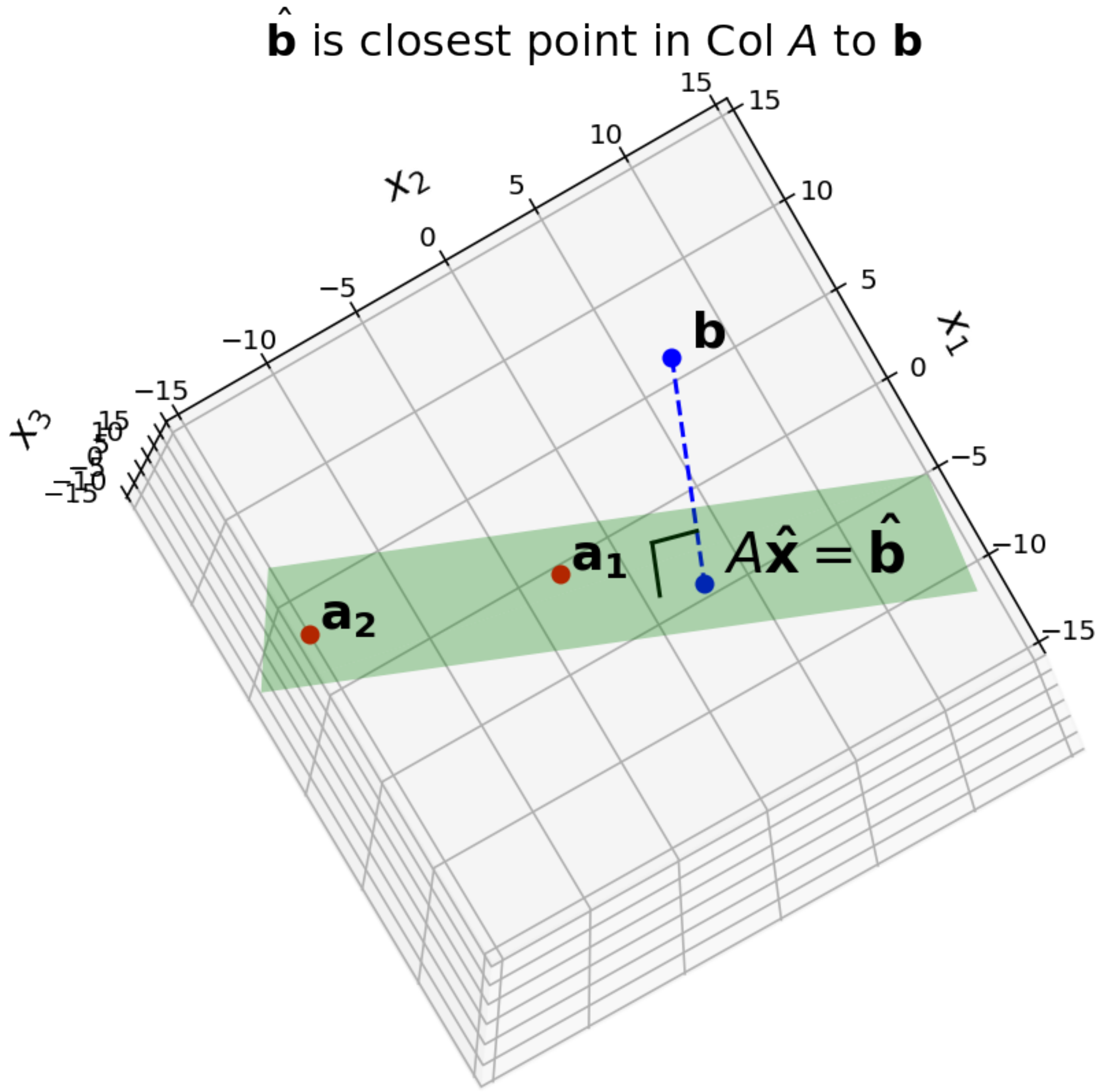
Question. Find a value α such that $\alpha\mathbf{u}$ is as close as possible to $\mathbf{y}$

That is, the distance $dist(\mathbf{y}, \alpha\mathbf{u}) = \|\mathbf{y} - \alpha\mathbf{u}\|$ is as small as possible

We need to generalize this to arbitrary matrix equations

The General Least Squares Problem

Figure 22.8

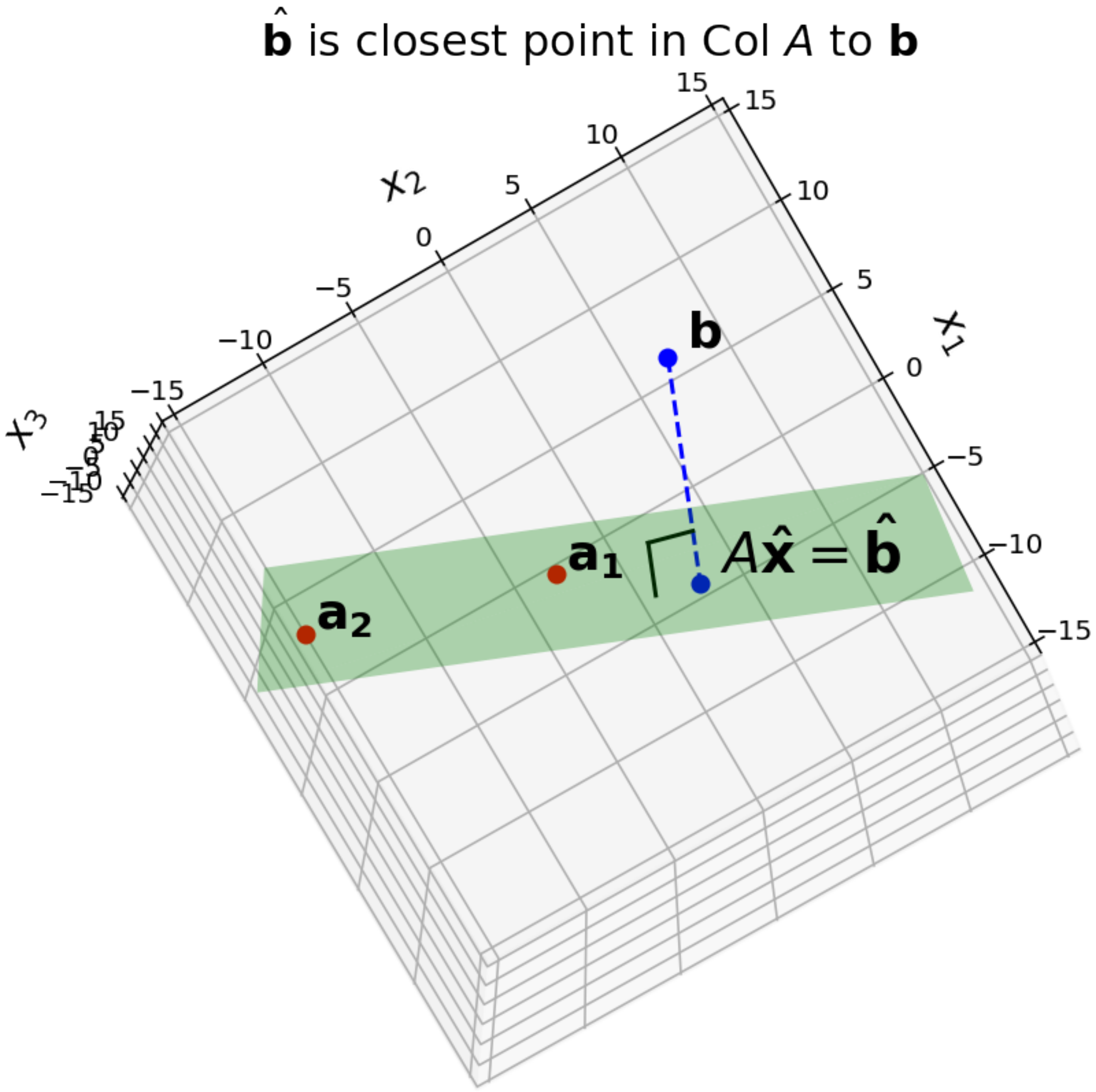


The General Least Squares Problem

Problem. Given a $m \times n$ matrix A and a vector $\mathbf{b}$ from $\mathbb{R}^m$, find a vector $\mathbf{x}$ in $\mathbb{R}^n$ which <u>minimizes</u>

$$dist(A\mathbf{x}, \mathbf{b}) = \|A\mathbf{x} - \mathbf{b}\|$$

Figure 22.8



The General Least Squares Problem

Problem. Given a $m \times n$ matrix A and a vector $\mathbf{b}$ from $\mathbb{R}^m$, find a vector $\mathbf{x}$ in $\mathbb{R}^n$ which <u>minimizes</u>

$$dist(A\mathbf{x}, \mathbf{b}) = \|A\mathbf{x} - \mathbf{b}\|$$

Find a vector $\mathbf{x}$ *which makes* $\|A\mathbf{x} - \mathbf{b}\|$ *as small as possible*

Figure 22.8

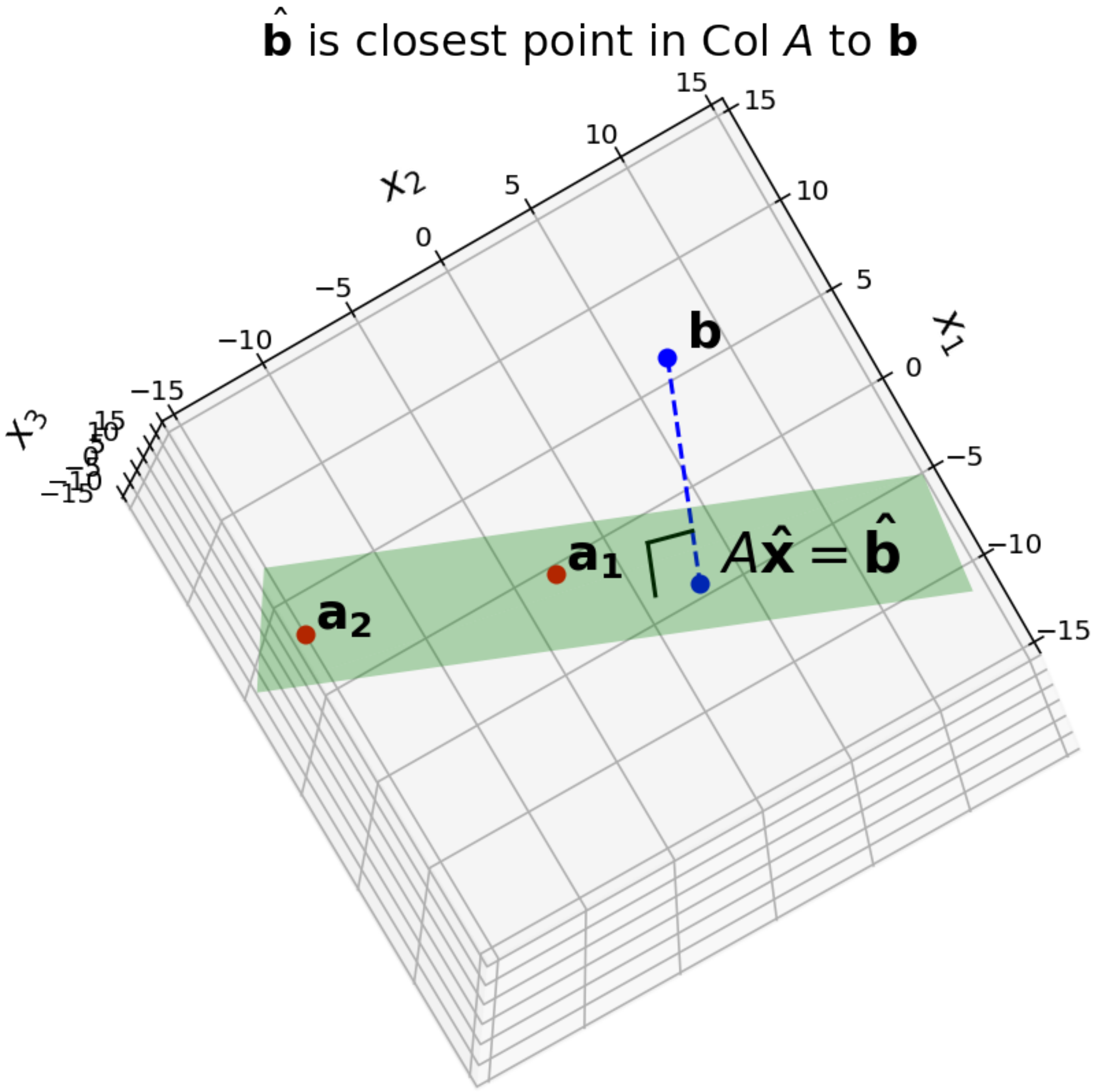
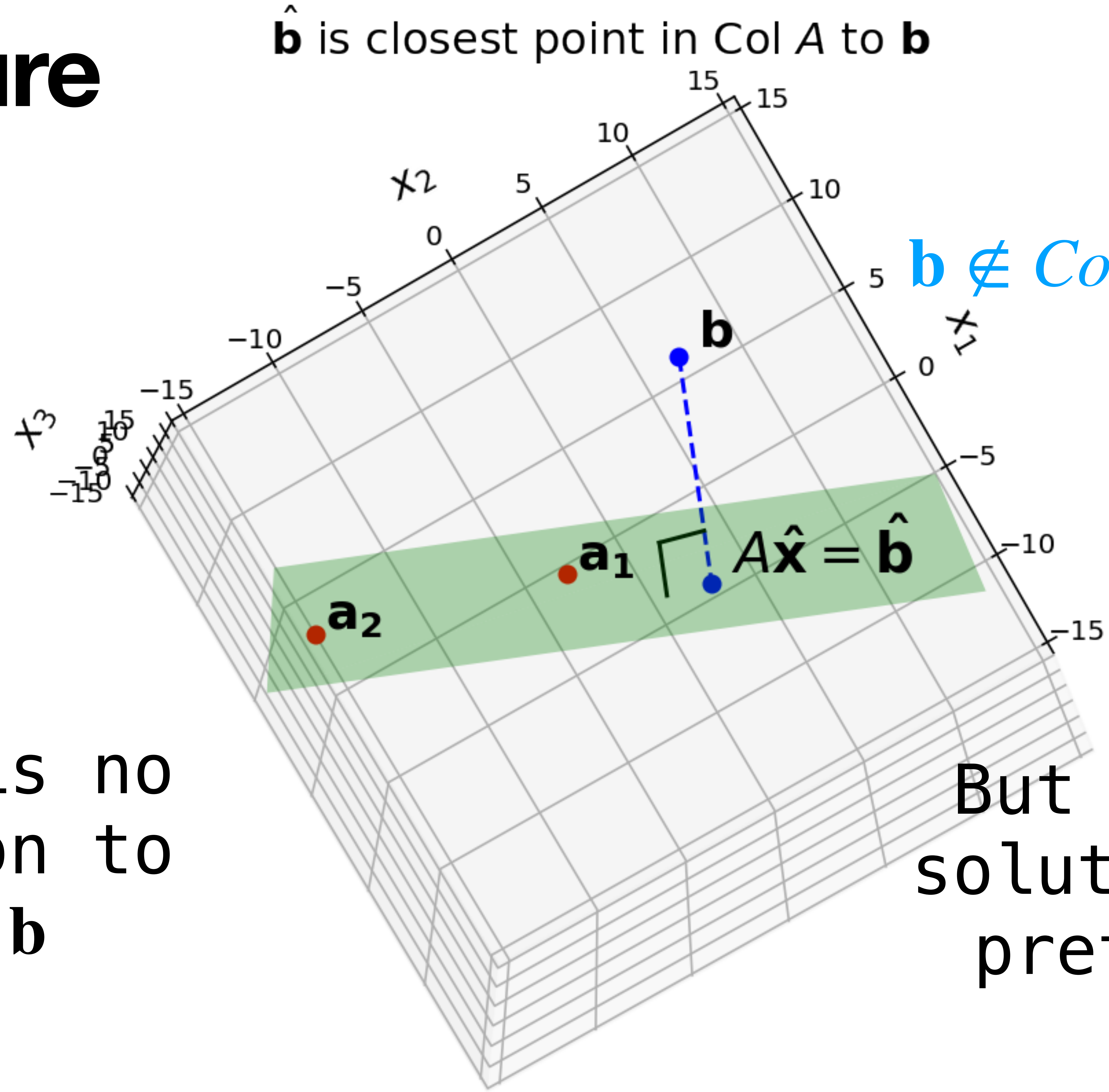


Figure 22.8

The Picture



$\mathbf{b} \notin Col(A)$

There is no solution to $A\mathbf{x} = \mathbf{b}$

But there's a solution that's pretty close

Sum of Squares

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \sum_{i=1}^{n} ((A\mathbf{x})_i - \mathbf{b}_i)^2$$

Sum of Squares

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \sum_{i=1}^{n} ((A\mathbf{x})_i - \mathbf{b}_i)^2$$

It is equivalent to minimize $\|A\mathbf{x} - \mathbf{b}\|^2$, which can be viewed as a **sum of squares**

Sum of Squares

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \sum_{i=1}^{n} ((A\mathbf{x})_i - \mathbf{b}_i)^2$$

It is equivalent to minimize $\|A\mathbf{x} - \mathbf{b}\|^2$, which can be viewed as a **sum of squares**

These things come up everywhere

Sum of Squares

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \sum_{i=1}^{n} ((A\mathbf{x})_i - \mathbf{b}_i)^2$$

It is equivalent to minimize $\|A\mathbf{x} - \mathbf{b}\|^2$, which can be viewed as a **sum of squares**

These things come up everywhere

(Advanced.) This error is everywhere differentiable, whereas $\sum_{i=1}^{n} |(A\mathbf{x})_i - b_i|$ *is not*

Least Squares Solution

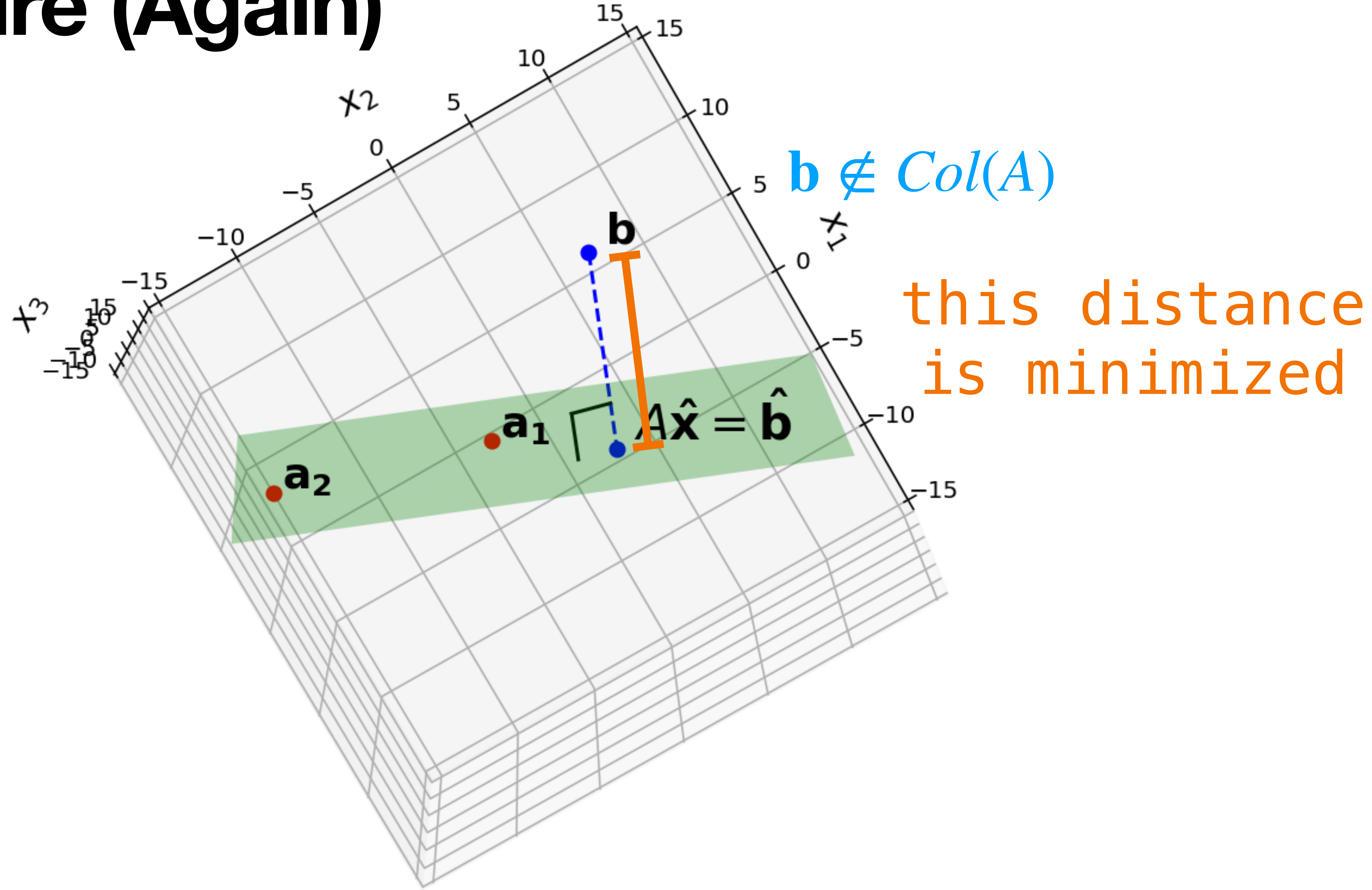
Definition. Given a $m \times n$ matrix A and a vector $\mathbf{b}$ in $\mathbb{R}^m$, a **least squares solution** of $A\mathbf{x} = \mathbf{b}$ is a vector $\hat{\mathbf{x}}$ from $\mathbb{R}^n$ such that

$$\|A\hat{\mathbf{x}} - \mathbf{b}\| \leq \|A\mathbf{x} - \mathbf{b}\|$$

for <u>any</u> $\mathbf{x}$ in $\mathbb{R}^n$

Again, $\|A\hat{\mathbf{x}} - \mathbf{b}\|$ is as small as possible

The Picture (Again)



Argmin

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|$$

Argmin

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|$$

Another way of framing this is via $\arg\min$

Argmin

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|$$

Another way of framing this is via $\arg\min$

Defintion. $\arg\min_{x\in X} f(x) = \hat{x}$ where $f(\hat{x}) = \min_{x\in X} f(x)$

Argmin

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|$$

Another way of framing this is via $\arg\min$

Defintion. $\arg\min_{x\in X} f(x) = \hat{x}$ where $f(\hat{x}) = \min_{x\in X} f(x)$

$\hat{x}$ is the ***arg**ument* that ***min**imizes* f

Argmin

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|$$

Another way of framing this is via $\arg\min$

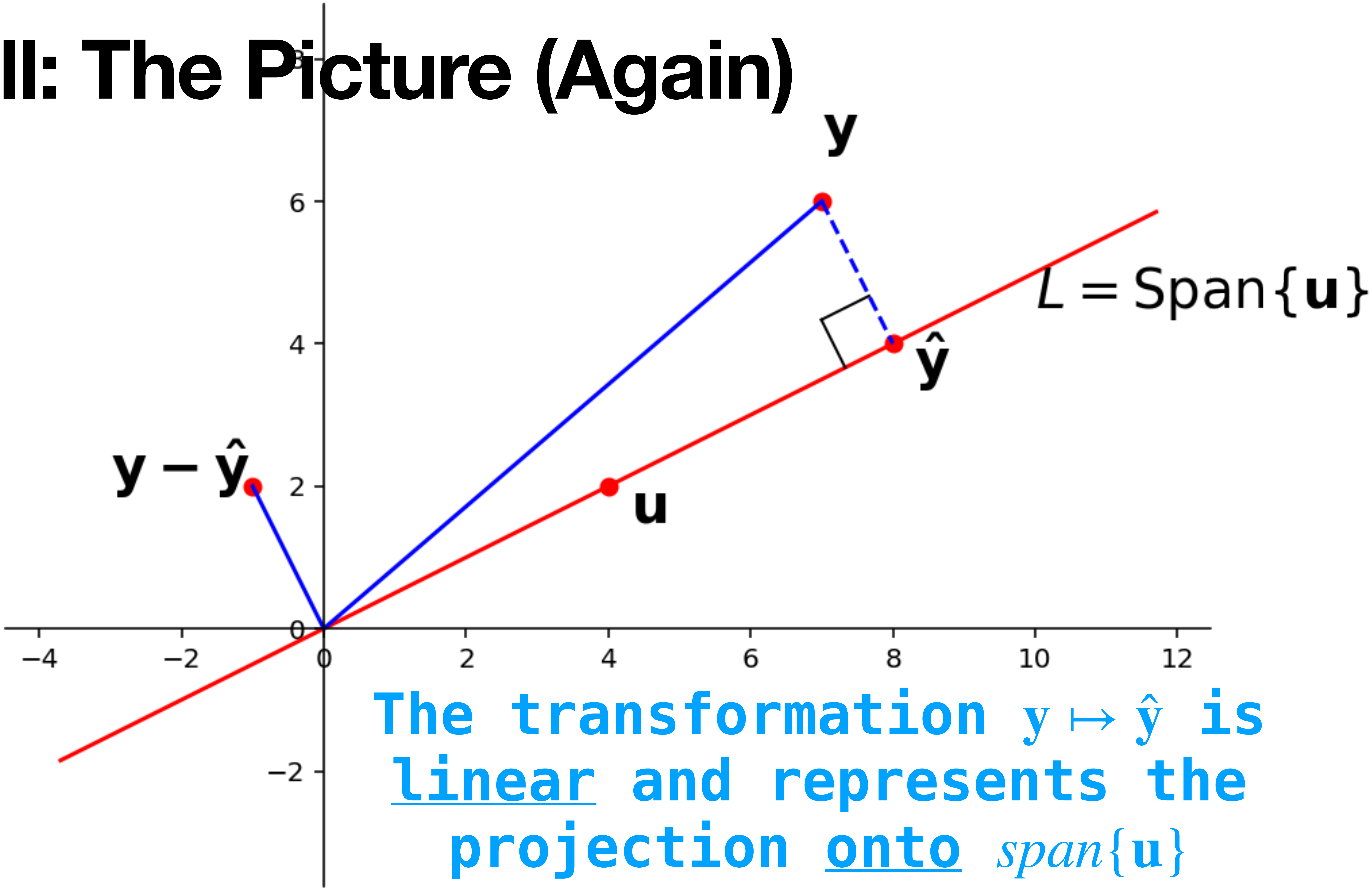
Defintion. $\arg\min_{x \in X} f(x) = \hat{x}$ where $f(\hat{x}) = \min_{x \in X} f(x)$

$\hat{x}$ is the ***arg****ument* that ***min****imizes* f

This is now an <u>optimization problem</u>

Solving the General Least Squares Problems

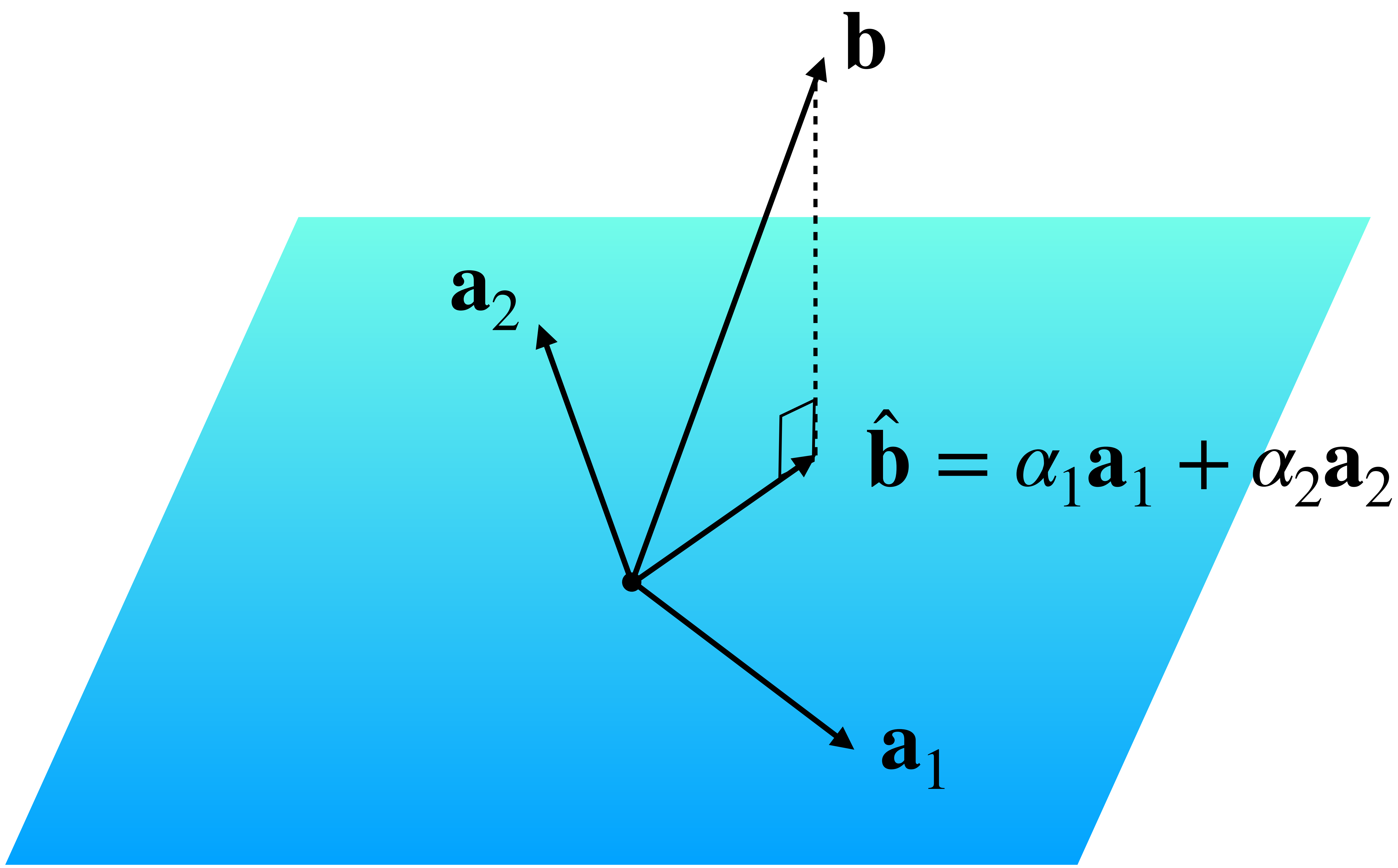
Recall: The Picture (Again)



The transformation $\mathbf{y} \mapsto \hat{\mathbf{y}}$ is linear and represents the projection onto $span\{\mathbf{u}\}$

Projects onto other Spans

The transformation $\mathbf{b} \mapsto \hat{\mathbf{b}}$ is the projection of $\mathbf{b}$ onto $\text{span}\{\mathbf{a}_1, \mathbf{a}_2\}$



The High Level Approach.

Question. Find a least squares solutions to $A = \mathbf{b}$

Solution.

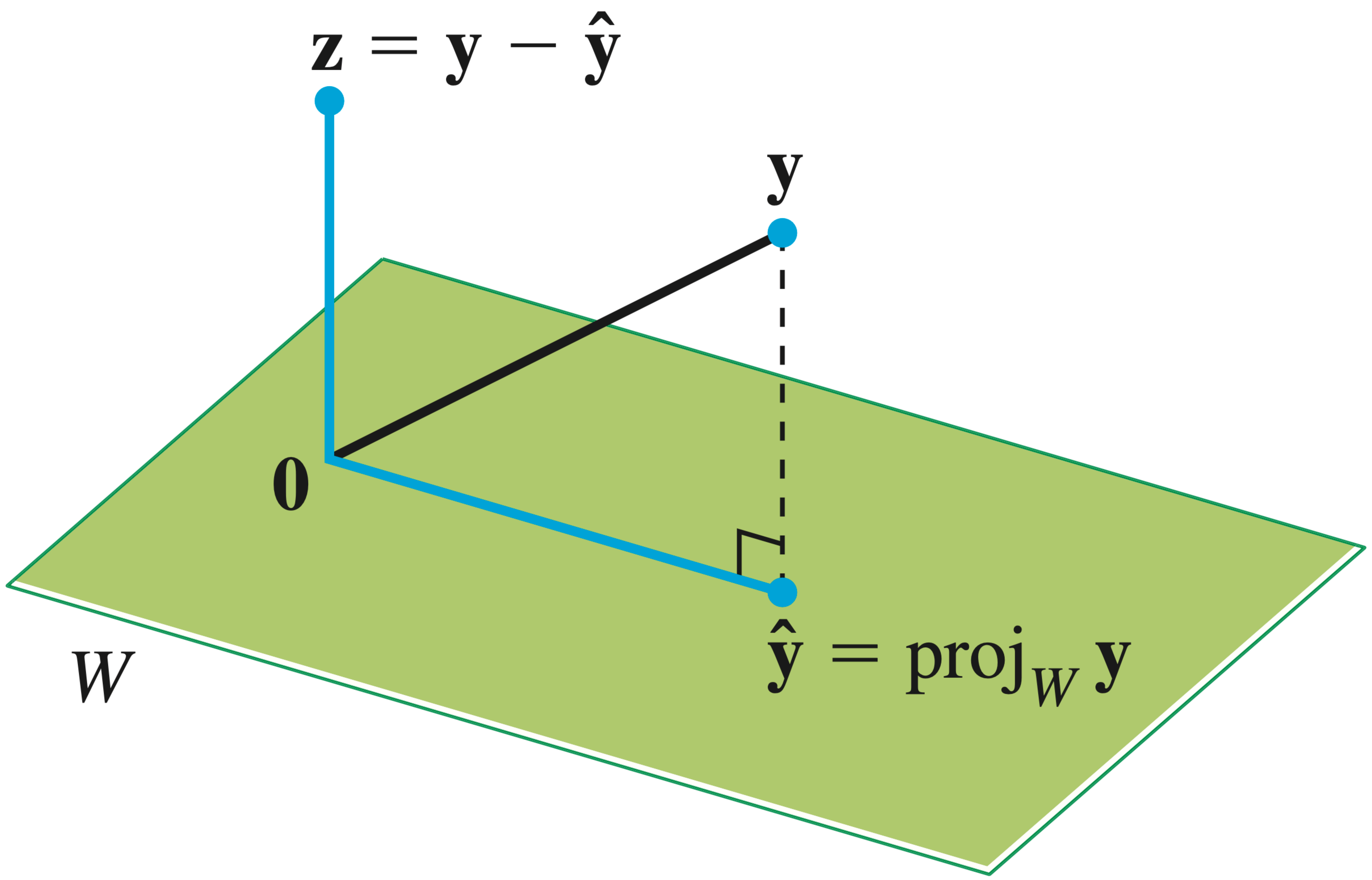
1. Find the closest point $\hat{\mathbf{b}}$ in $Col(A)$ to $\mathbf{b}$
2. Solve the equation $A\mathbf{x} = \hat{\mathbf{b}}$ instead

Orthogonal Decomposition Theorem

Theorem. Let W be a subspace of $\mathbb{R}^n$. Every vector $\mathbf{y}$ in $\mathbb{R}^n$ can be written <u>uniquely</u> as

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{z}$$

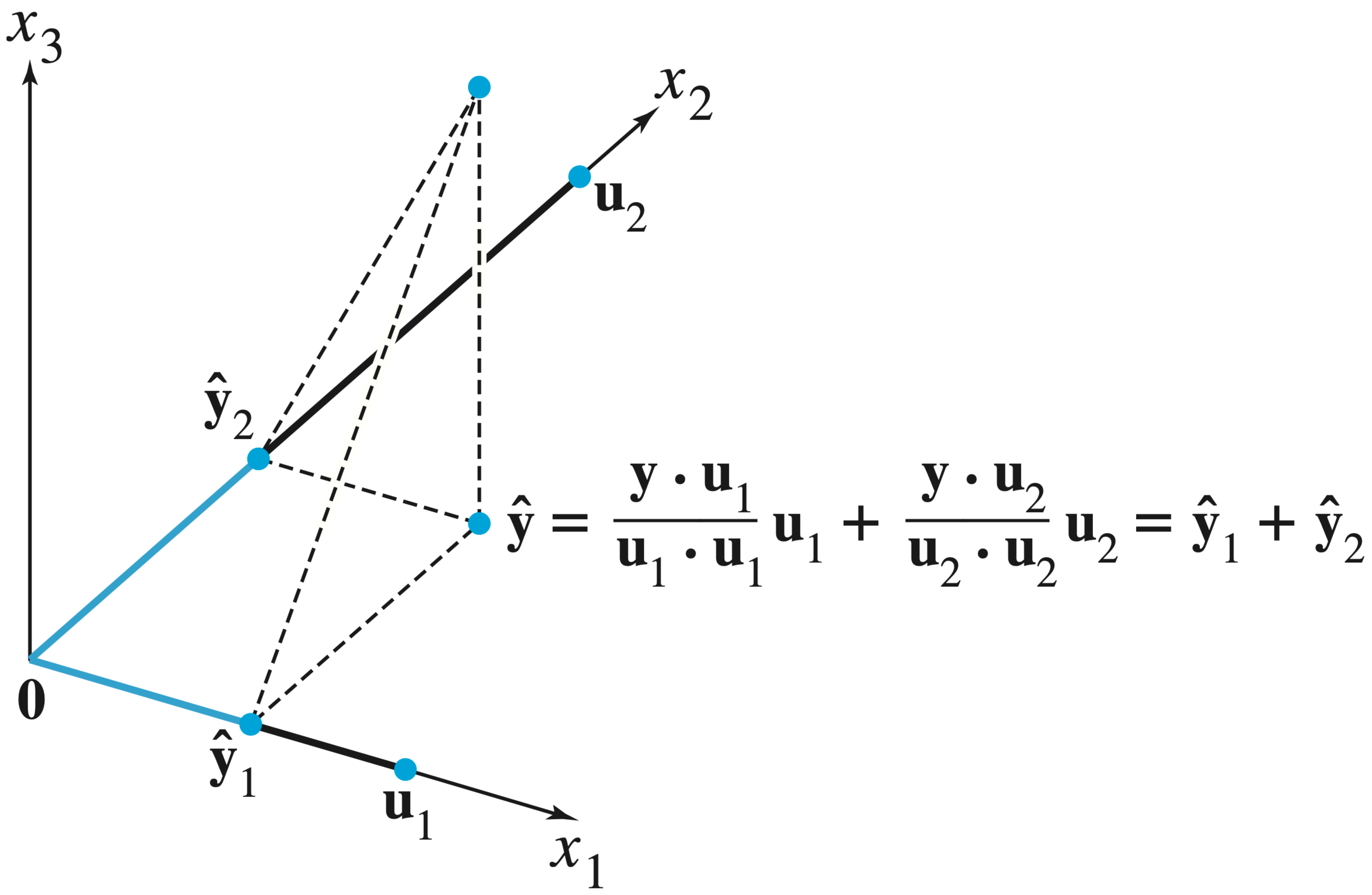
where $\hat{\mathbf{y}} \in W$ and $\mathbf{z}$ is orthogonal to every vector in W



Projection via Orthogonal Bases

We can determine $\hat{\mathbf{y}}$ by projecting onto an orthogonal basis

Every subspace has an orthogonal basis (we won't prove this)



Linear Algebra and its Applications, Lay, Lay, McDonald

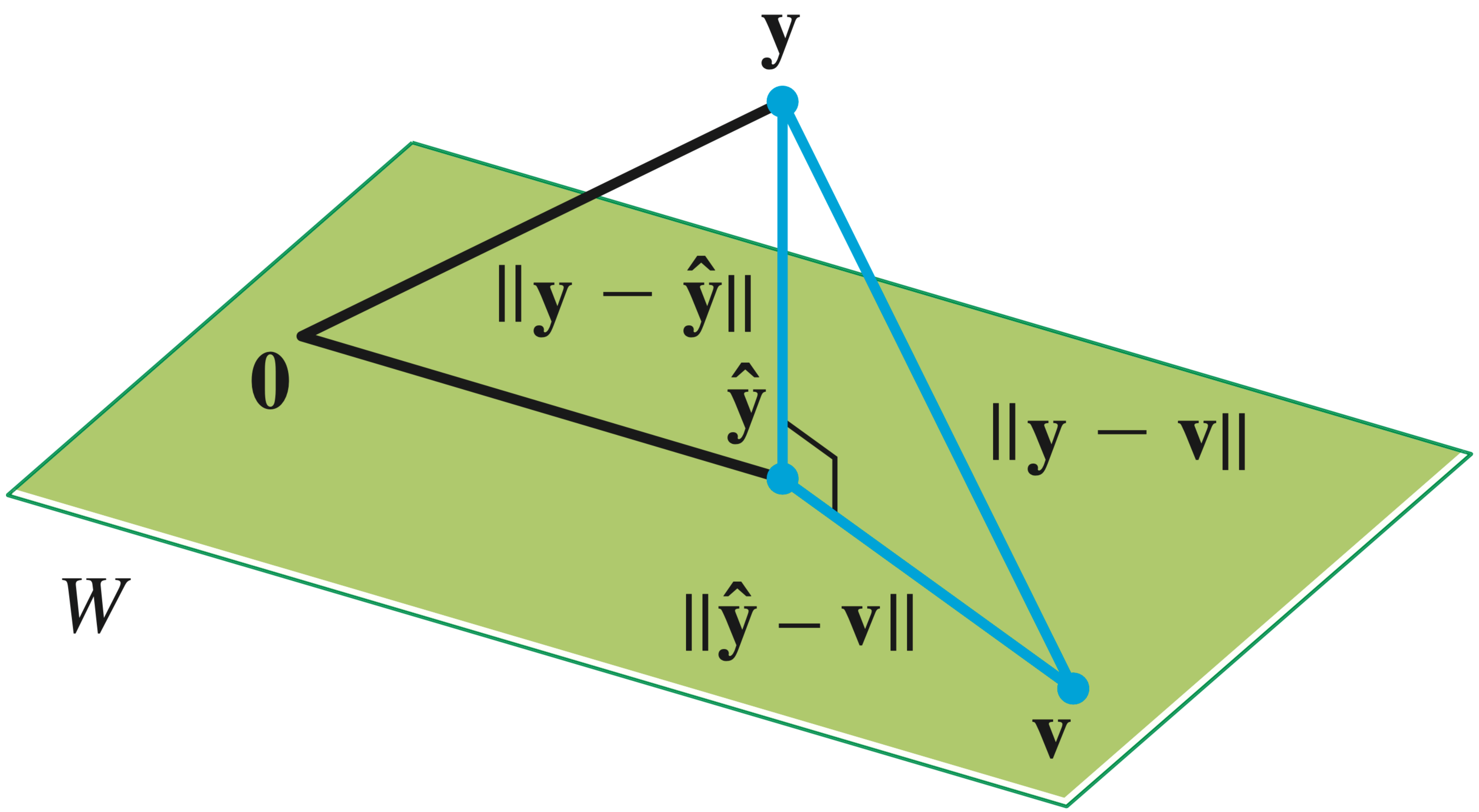
The Best-Approximation Theorem

Theorem. Let W be a subspace of $\mathbb{R}^n$, and let $\hat{\mathbf{y}}$ be the orthogonal projection of $\mathbf{y}$ onto W Then

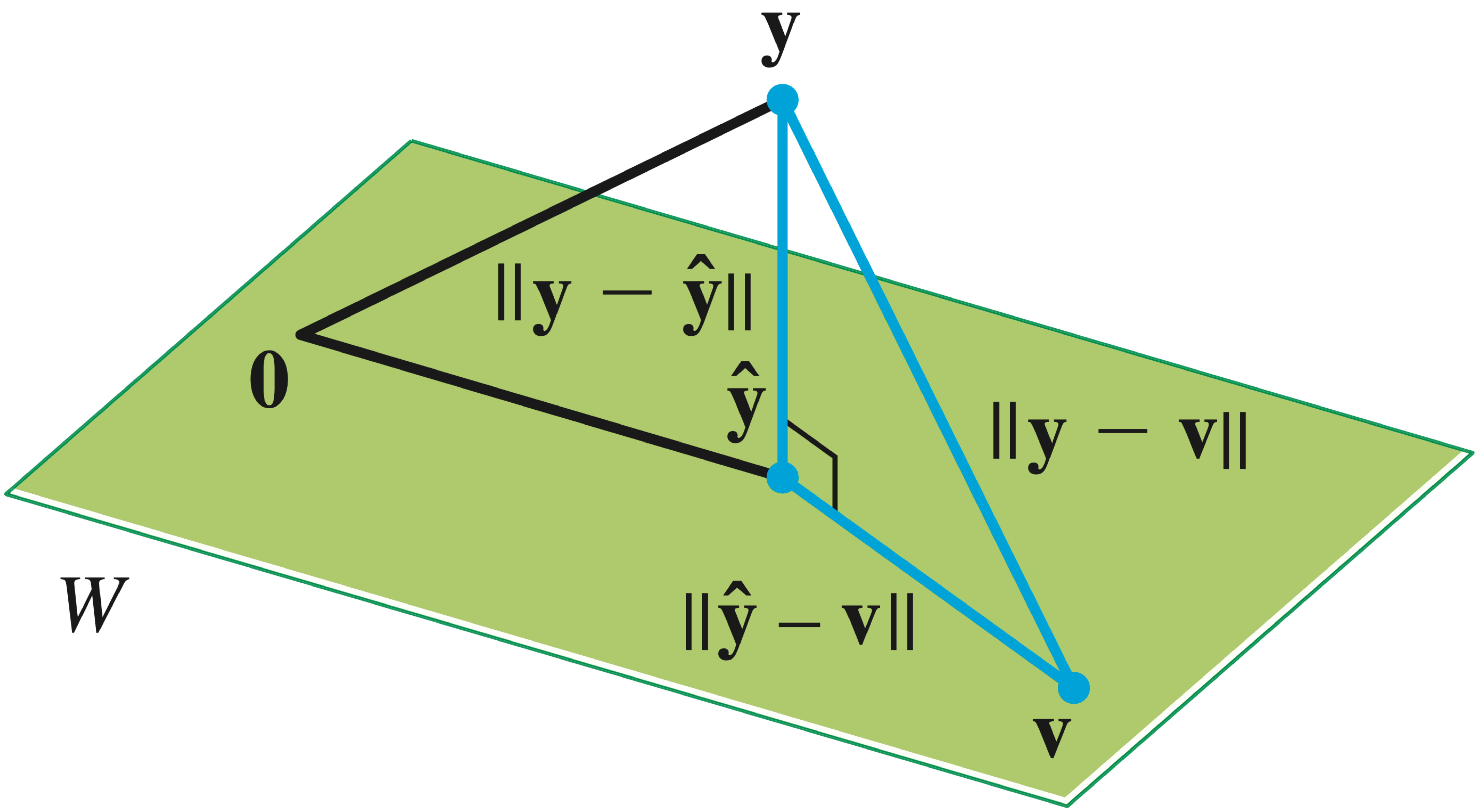
$$\|\mathbf{y} - \hat{\mathbf{y}}\| \leq \|\mathbf{y} - \mathbf{w}\|$$

for <u>any</u> vector $\mathbf{w}$ in W

$\hat{\mathbf{y}}$ is the closest point in W to $\mathbf{y}$

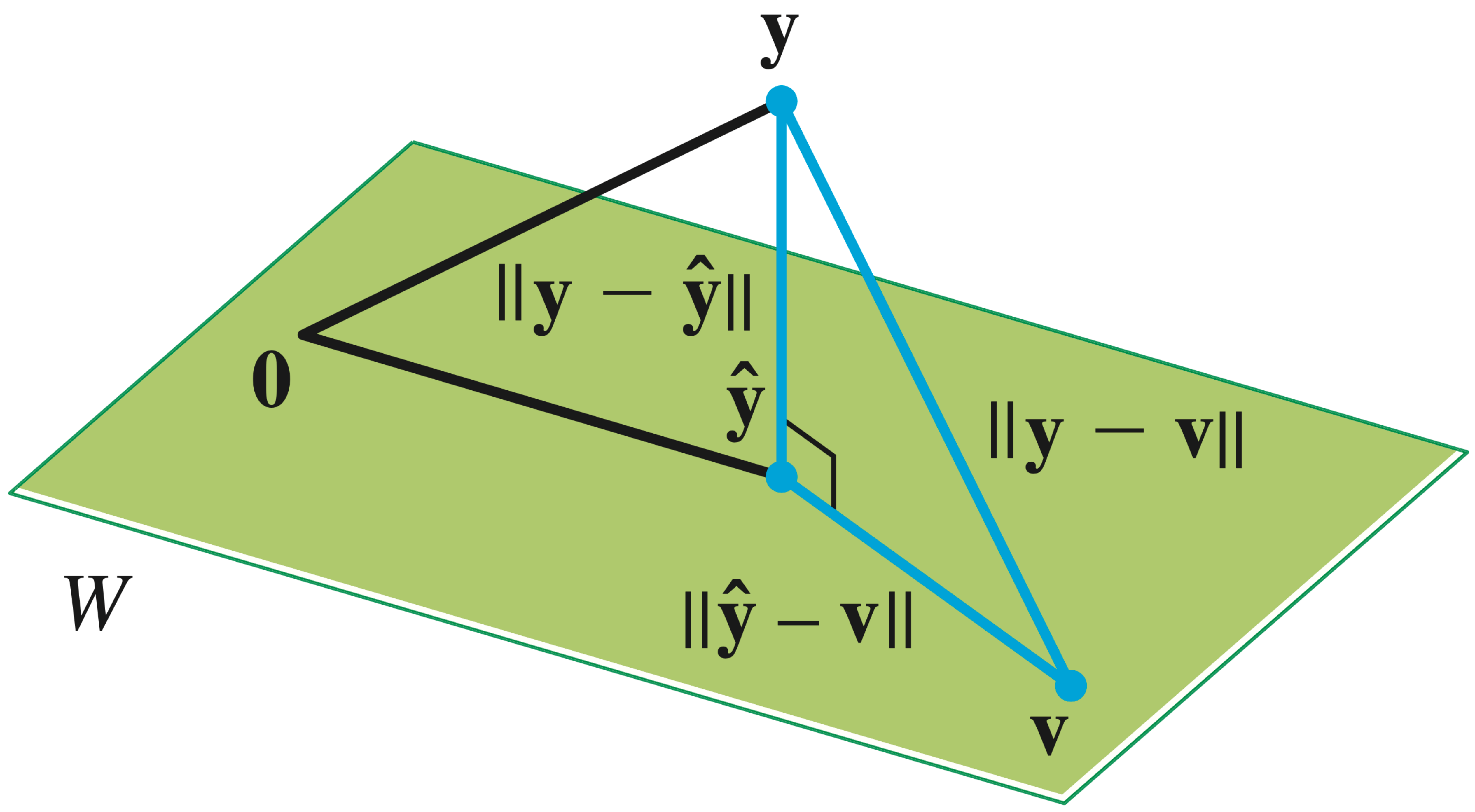


Proof by Inspection

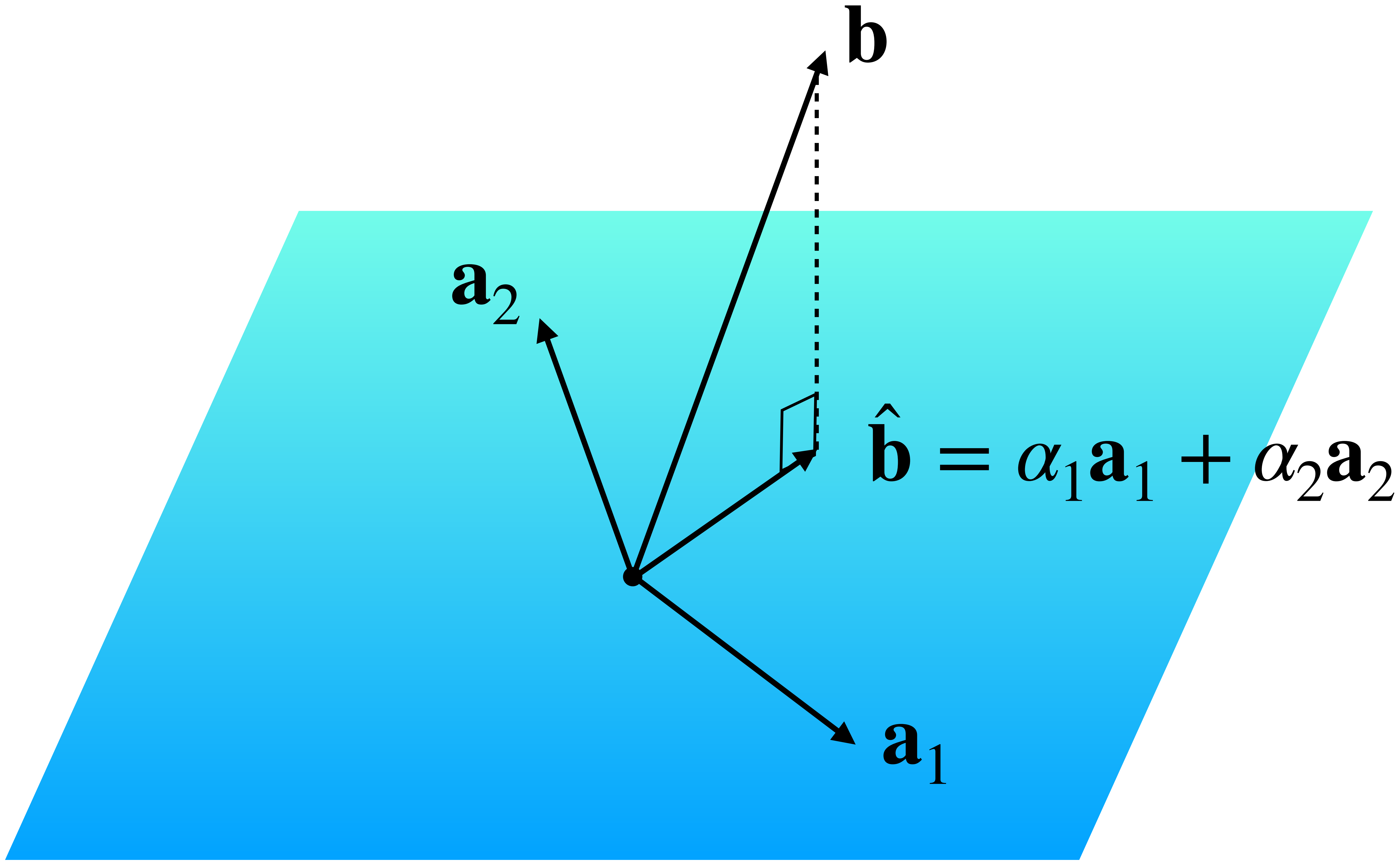


Proof by Algebra

Verify:



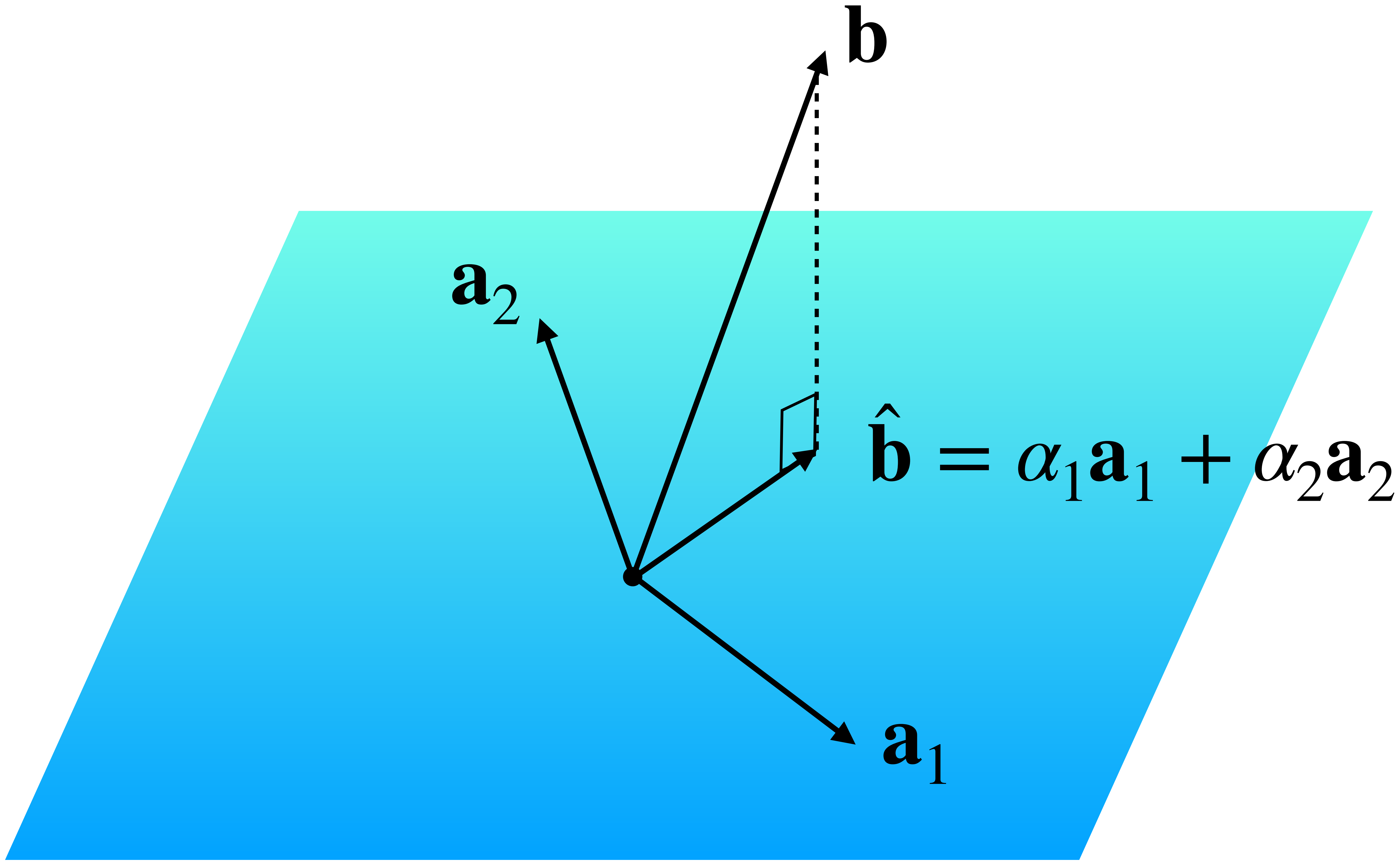
The Point



$\mathbf{b}$
$\mathbf{a}_2$
$\hat{\mathbf{b}} = \alpha_1\mathbf{a}_1 + \alpha_2\mathbf{a}_2$
$\mathbf{a}_1$

The Point

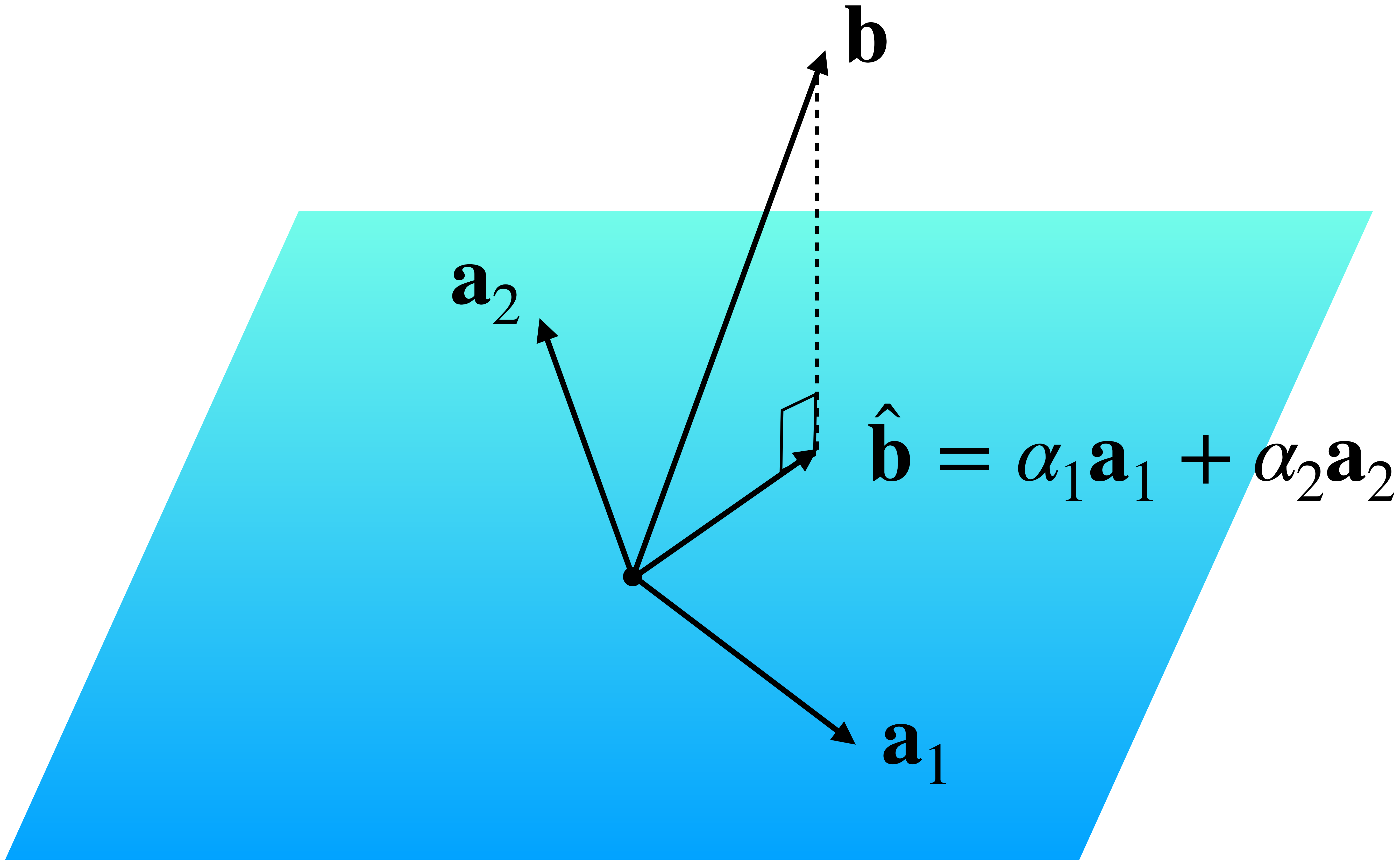
$\hat{\mathbf{b}}$ is in $Col(A)$ so $A\mathbf{x} = \hat{\mathbf{b}}$ has a solution



The Point

$\hat{\mathbf{b}}$ is in $Col(A)$ so $A\mathbf{x} = \hat{\mathbf{b}}$ **has a solution**

At this point, we could call it a day:

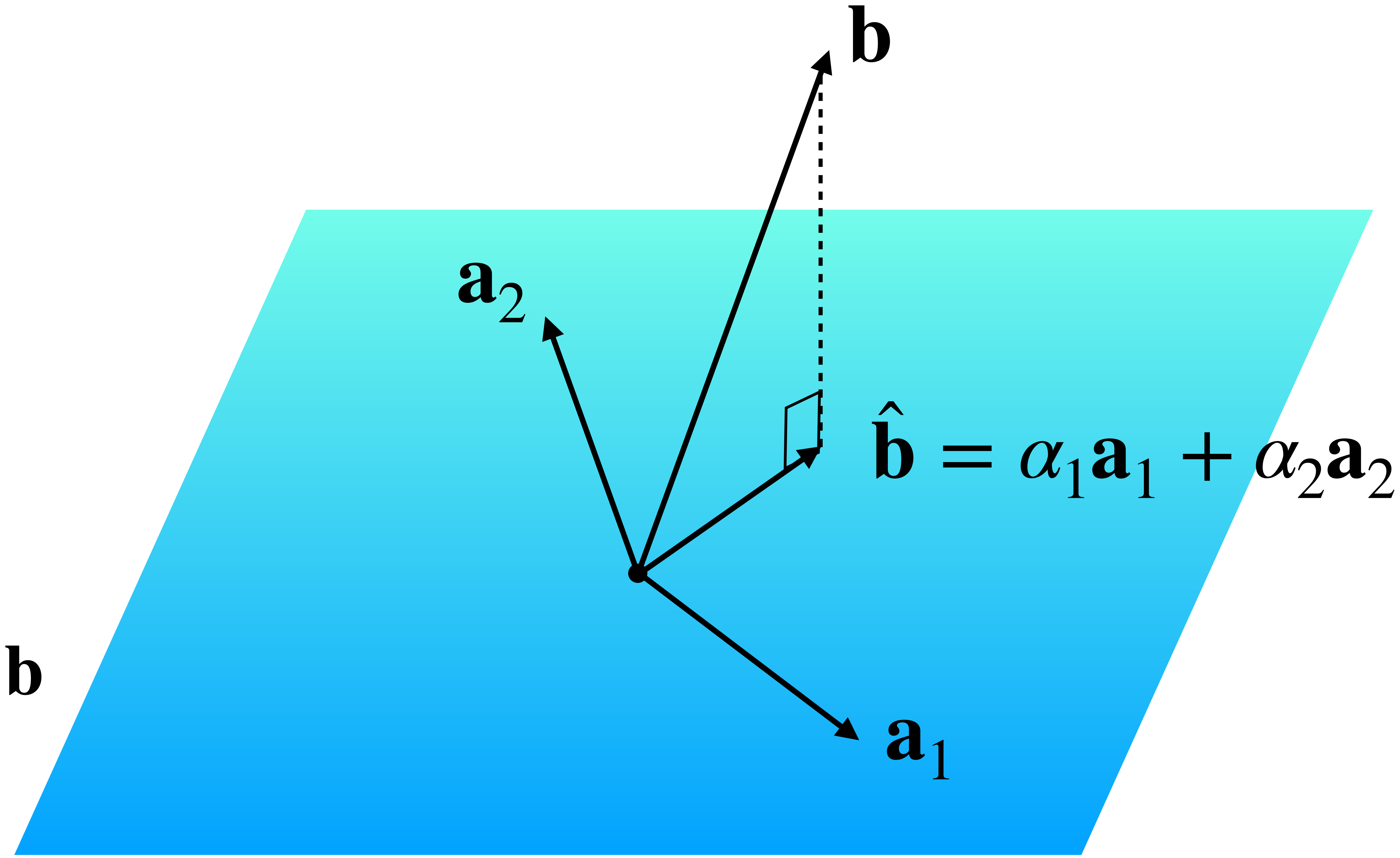


The Point

$\hat{\mathbf{b}}$ is in $Col(A)$ so $A\mathbf{x} = \hat{\mathbf{b}}$ **has a solution**

At this point, we could call it a day:

Question. Find a least squares solution to $A\mathbf{x} = \mathbf{b}$



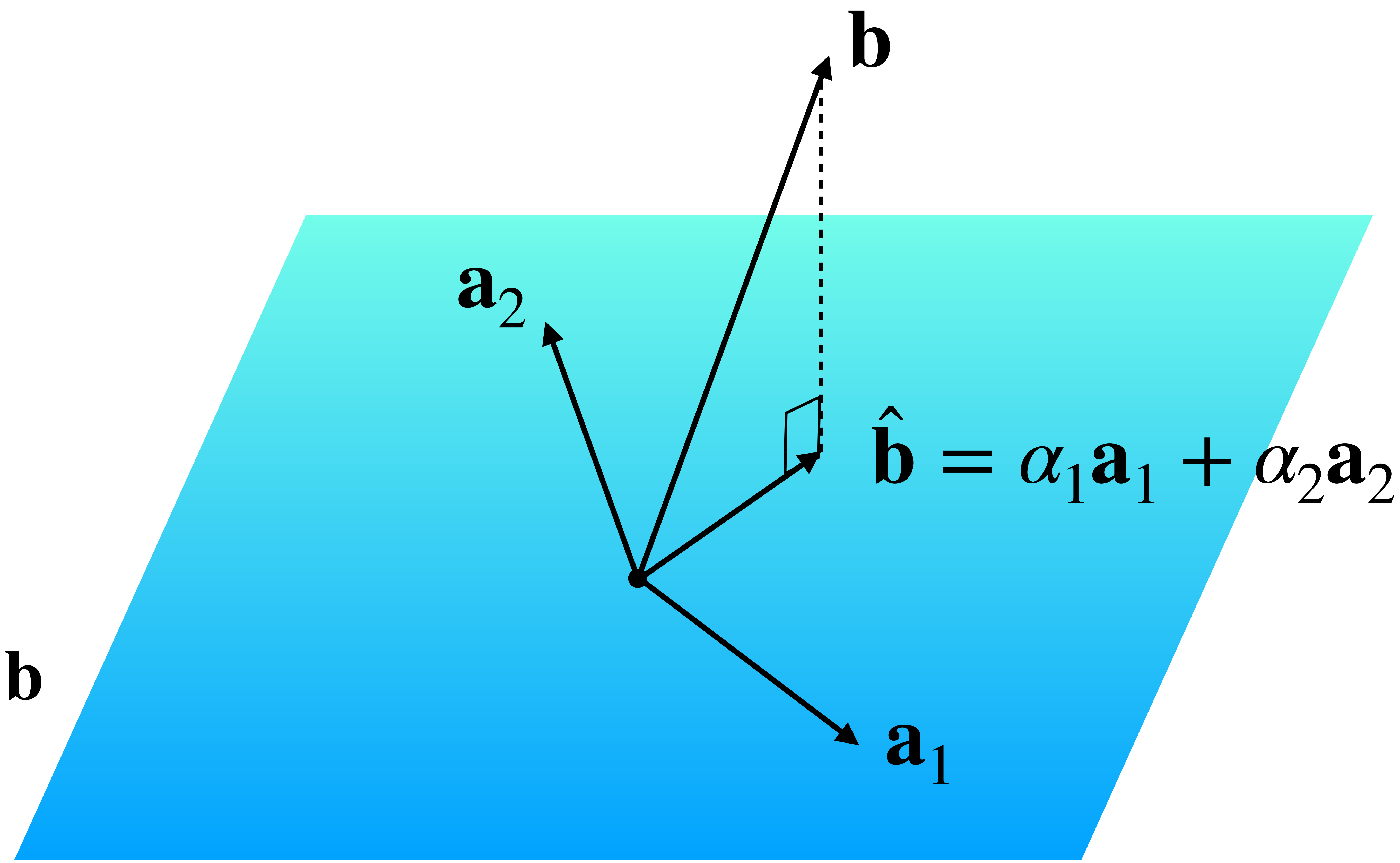
The Point

$\hat{\mathbf{b}}$ is in $Col(A)$ so $A\mathbf{x} = \hat{\mathbf{b}}$ **has a solution**

At this point, we could call it a day:

Question. Find a least squares solution to $A\mathbf{x} = \mathbf{b}$

Solution. Find $\hat{\mathbf{b}}$, then solve $A\mathbf{x} = \hat{\mathbf{b}}$



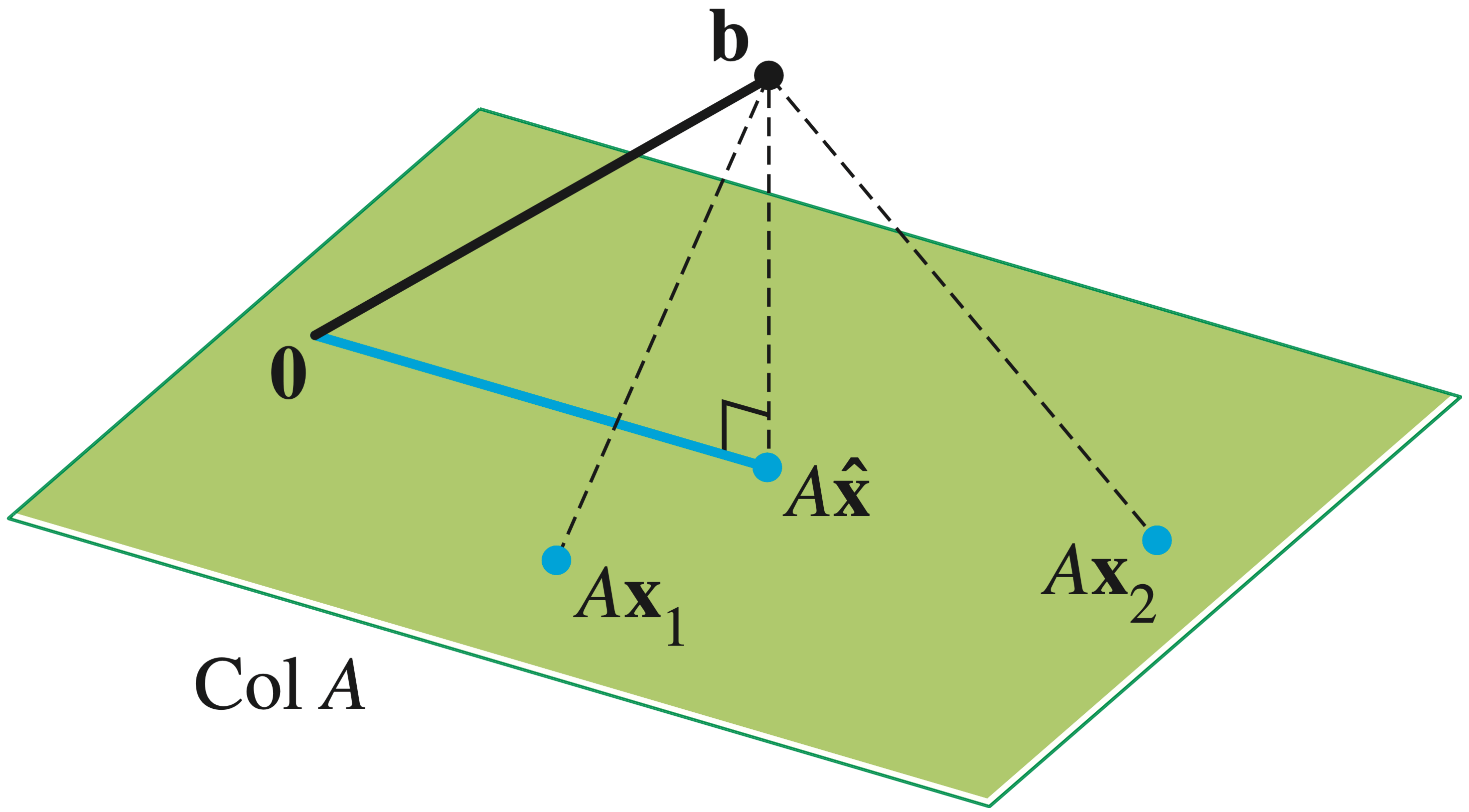
Example

$$\begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 0 & 0 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 4 \\ 1 \\ 4 \end{bmatrix}$$

Let's determine the least squares solution for the above system:

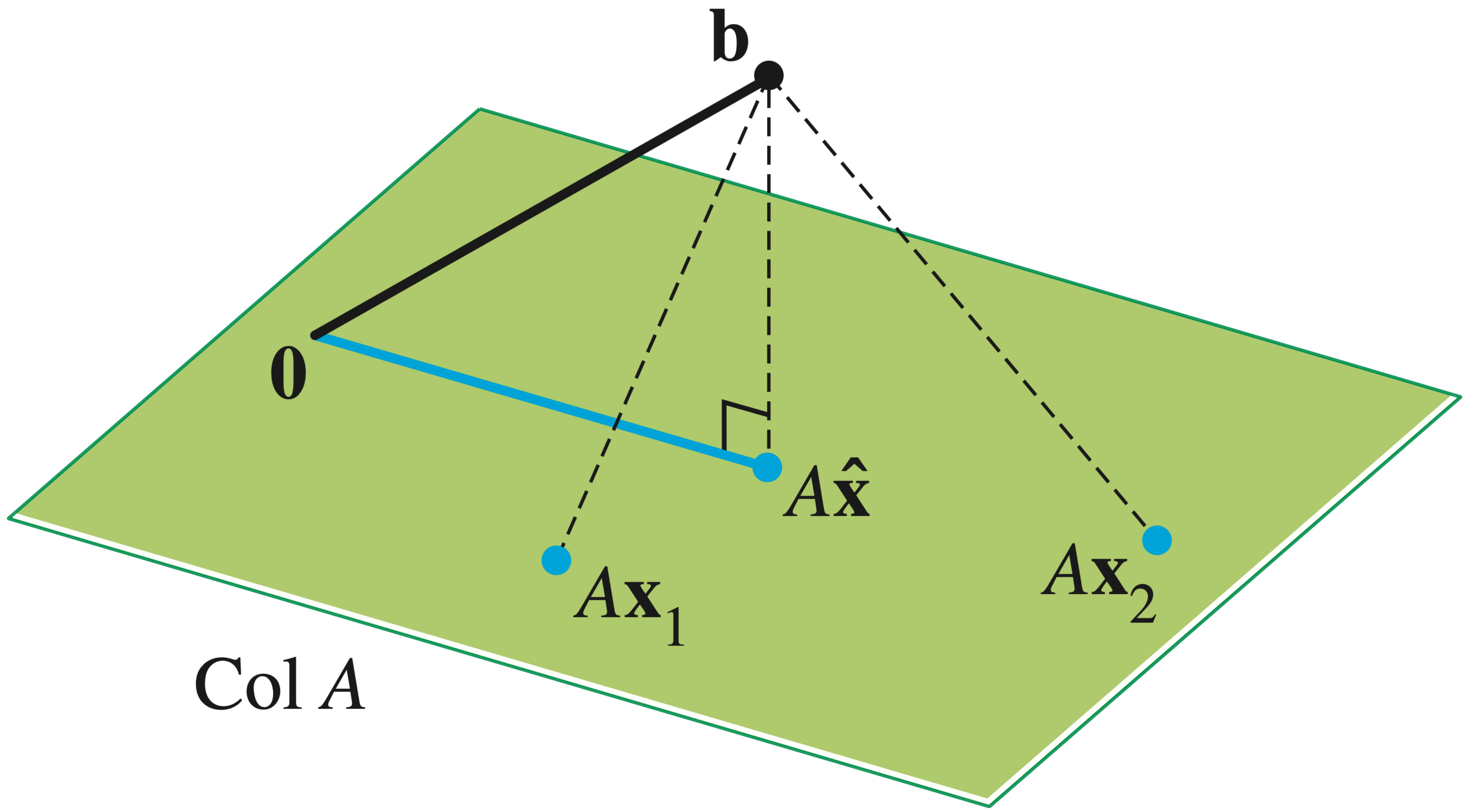
The Normal Equations

A Couple Observations



A Couple Observations

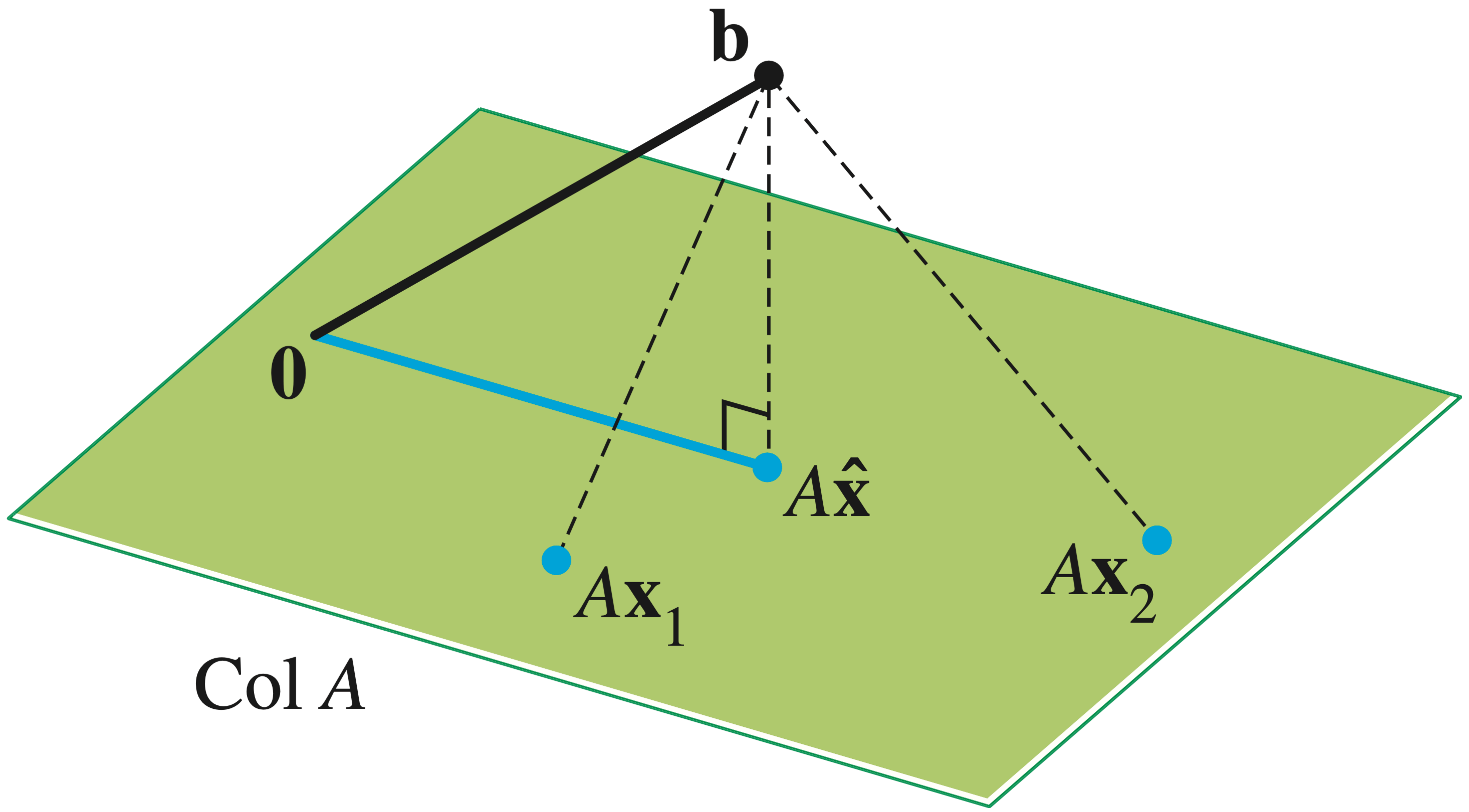
Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$



A Couple Observations

Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$

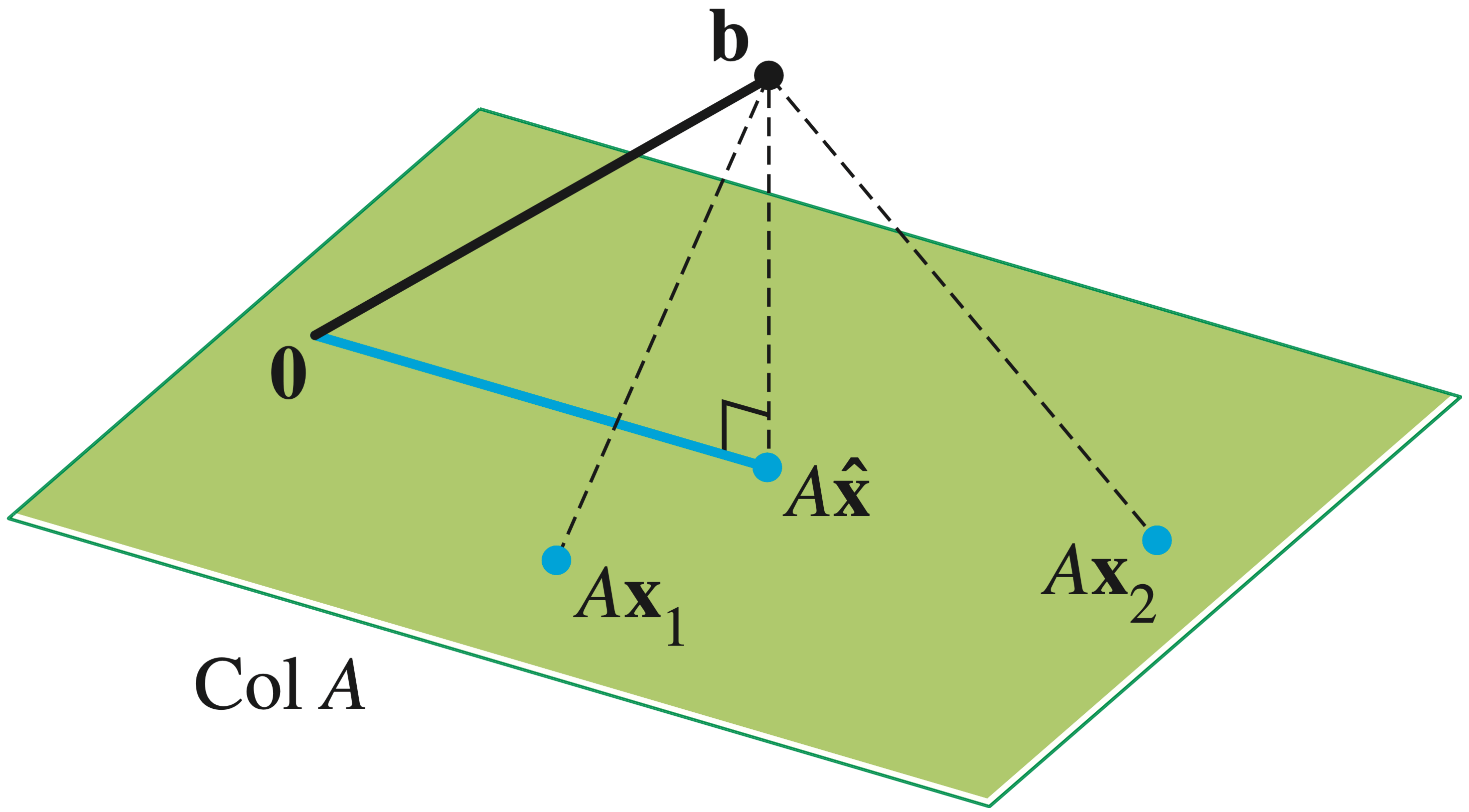
- $\hat{\mathbf{b}} - \mathbf{b}$ is orthogonal to $Col(A)$



A Couple Observations

Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$

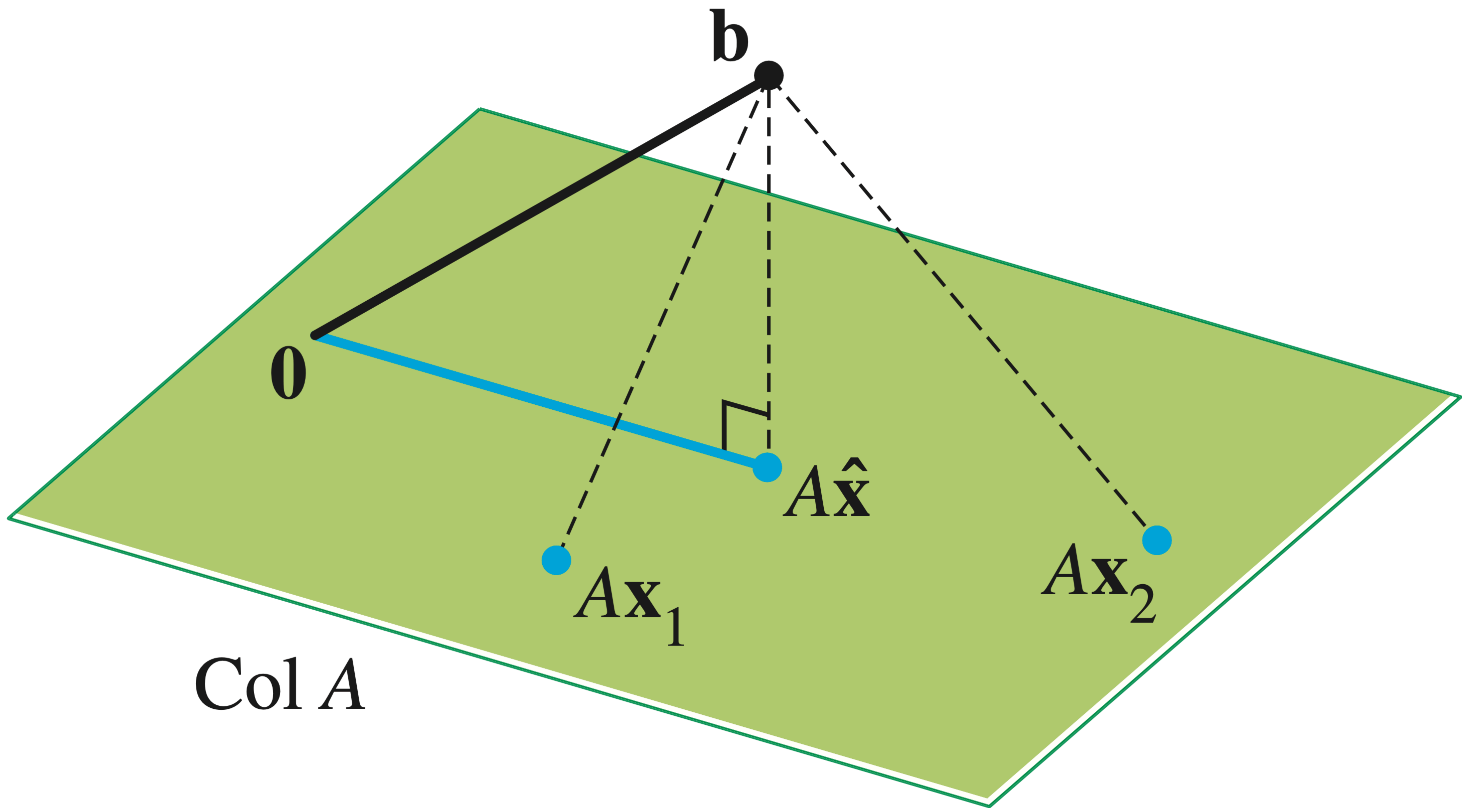
- $\hat{\mathbf{b}} - \mathbf{b}$ is orthogonal to $Col(A)$
- $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to $Col(A)$



A Couple Observations

Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$

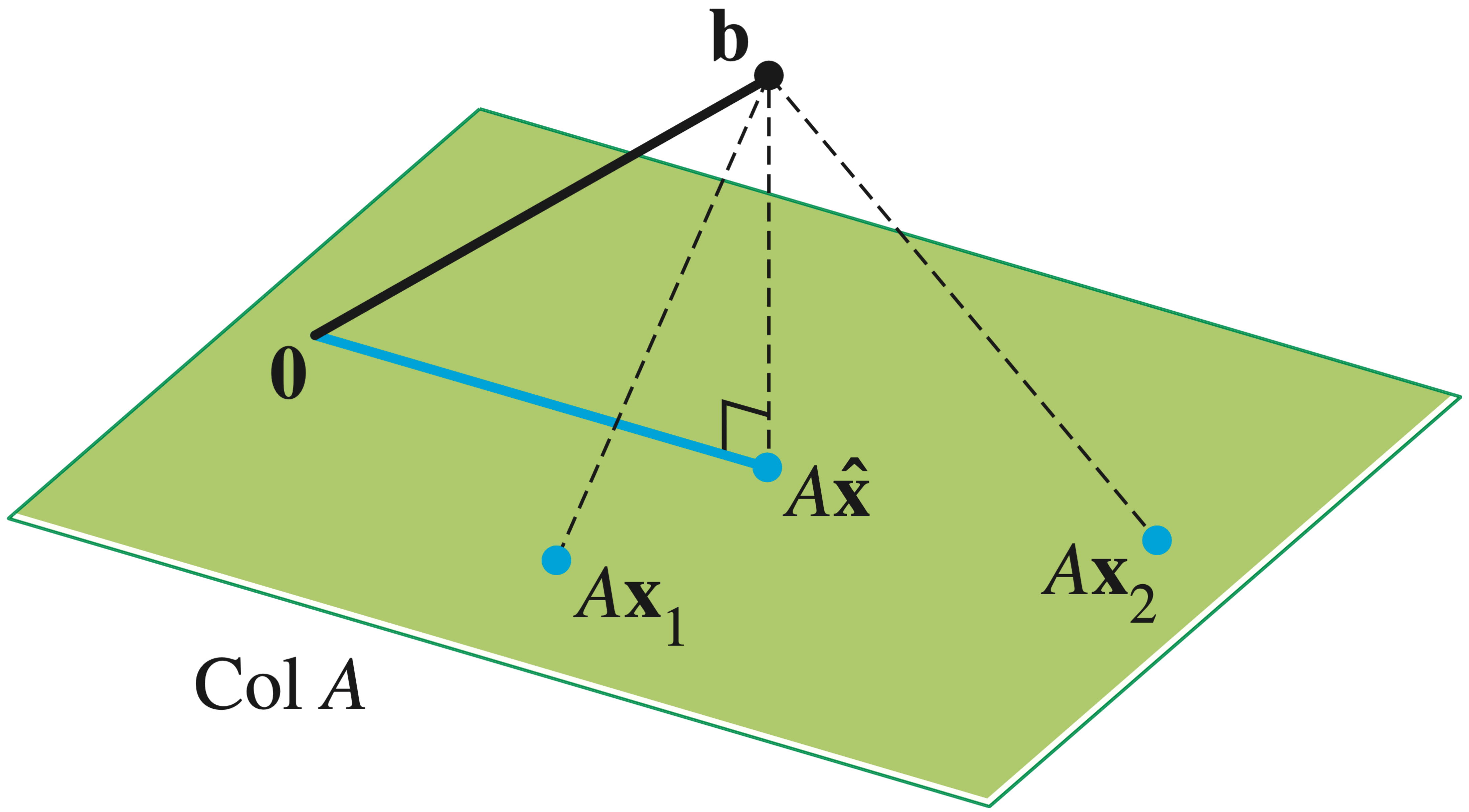
- $\hat{\mathbf{b}} - \mathbf{b}$ is orthogonal to $Col(A)$
- $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to $Col(A)$
- If $A = [\mathbf{a}_1 \;\; \mathbf{a}_2 \;\; \dots \;\; \mathbf{a}_n]$ then $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to each $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$



A Couple Observations

Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$

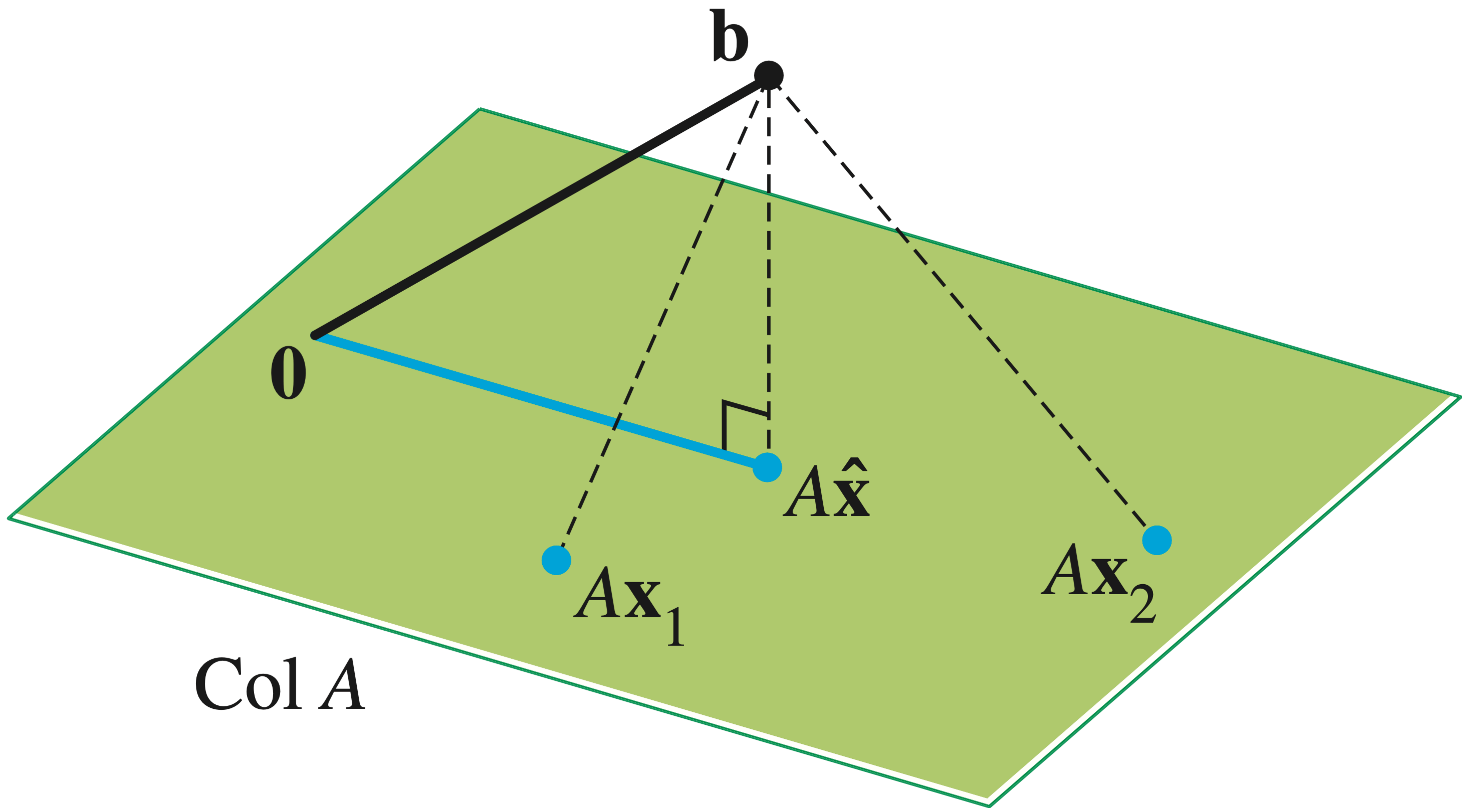
- $\hat{\mathbf{b}} - \mathbf{b}$ is orthogonal to $Col(A)$
- $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to $Col(A)$
- If $A = [\mathbf{a}_1 \ \ \mathbf{a}_2 \ \ \dots \ \ \mathbf{a}_n]$ then $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to each $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$
- $\mathbf{a}_i^T(A\hat{\mathbf{x}} - \mathbf{b}) = 0$



A Couple Observations

Suppose that $\hat{\mathbf{x}}$ is a least squares solution to A, so $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$

- $\hat{\mathbf{b}} - \mathbf{b}$ is orthogonal to $Col(A)$
- $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to $Col(A)$
- If $A = [\mathbf{a}_1 \;\; \mathbf{a}_2 \;\; \dots \;\; \mathbf{a}_n]$ then $A\hat{\mathbf{x}} - \mathbf{b}$ is orthogonal to each $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$
- $\mathbf{a}_i^T(A\hat{\mathbf{x}} - \mathbf{b}) = 0$
- $A^T(A\hat{\mathbf{x}} - \mathbf{b}) = \mathbf{0}$



A bit more magic

Let's simplify $A^T(A\hat{\mathbf{x}} - \mathbf{b})$:

The Normal Equations

The Normal Equations

Theorem. The set of least-squares solutions of $A\mathbf{x} = \mathbf{b}$ is the same as the set of solutions to

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

The Normal Equations

Theorem. The set of least-squares solutions of $A\mathbf{x} = \mathbf{b}$ is the same as the set of solutions to

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

In particular, this set of solutions is nonempty

The Normal Equations

Theorem. The set of least-squares solutions of $A\mathbf{x} = \mathbf{b}$ is the same as the set of solutions to

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

In particular, this set of solutions is nonempty

(We just showed that if $\hat{\mathbf{x}}$ is a least squares solution then $A^T A\hat{\mathbf{x}} = A^T\mathbf{b}$)

Example

$$A = \begin{bmatrix} 4 & 0 \\ 0 & 2 \\ 1 & 1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 2 \\ 0 \\ 11 \end{bmatrix}$$

Let's find the normal equations for $A\mathbf{x} = \mathbf{b}$:

Example

$$\begin{bmatrix} 17 & 1 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 19 \\ 11 \end{bmatrix}$$

Let's solve the normal equations for $A\mathbf{x} = \mathbf{b}$:

Example

$$\begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 0 & 0 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 4 \\ 1 \\ 4 \end{bmatrix}$$

```
Let's do it again...
```

Unique Least Squares Solutions

Question (Conceptual)

Is a least squares solution unique?

Answer: No

Remember that if $\mathbf{b} \in Col(A)$ then $\hat{\mathbf{b}} = \mathbf{b}$ and then we're asking if $A\mathbf{x} = \mathbf{b}$ has a unique solution for any choice of A

When is there a unique solution?

The least squares method gives us to find an approximate solution when there is no exact solution

But it doesn't help us choose a solution in the case that there are many

Practically Speaking

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Practically Speaking

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

NumPy chooses the shortest vector

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Practically Speaking

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

`NumPy chooses the shortest vector`

`(why?...)`

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Unique Least Squares Solutions

Theorem. For a $m \times n$ matrix A the following are equivalent:

- » $A\mathbf{x} = \mathbf{b}$ has a unique least squares solution for any choice of $\mathbf{b}$
- » The columns of A are linearly independent
- » $A^T A$ is invertible

Unique Least Squares Solutions

$$\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}$$

If A has linearly independent columns, then its unique least squares solution is defined as above:

Projecting onto a subspace

$$\hat{\mathbf{b}} = A\hat{\mathbf{x}} = A(A^TA)^{-1}A^T\mathbf{b}$$

If the columns of A are linearly independent, then **they form a basis**

Said another way: if $\mathscr{B}$ is a basis, then we can construct a matrix A whose columns are the vectors in $\mathscr{B}$

This means we can find arbitrary projections