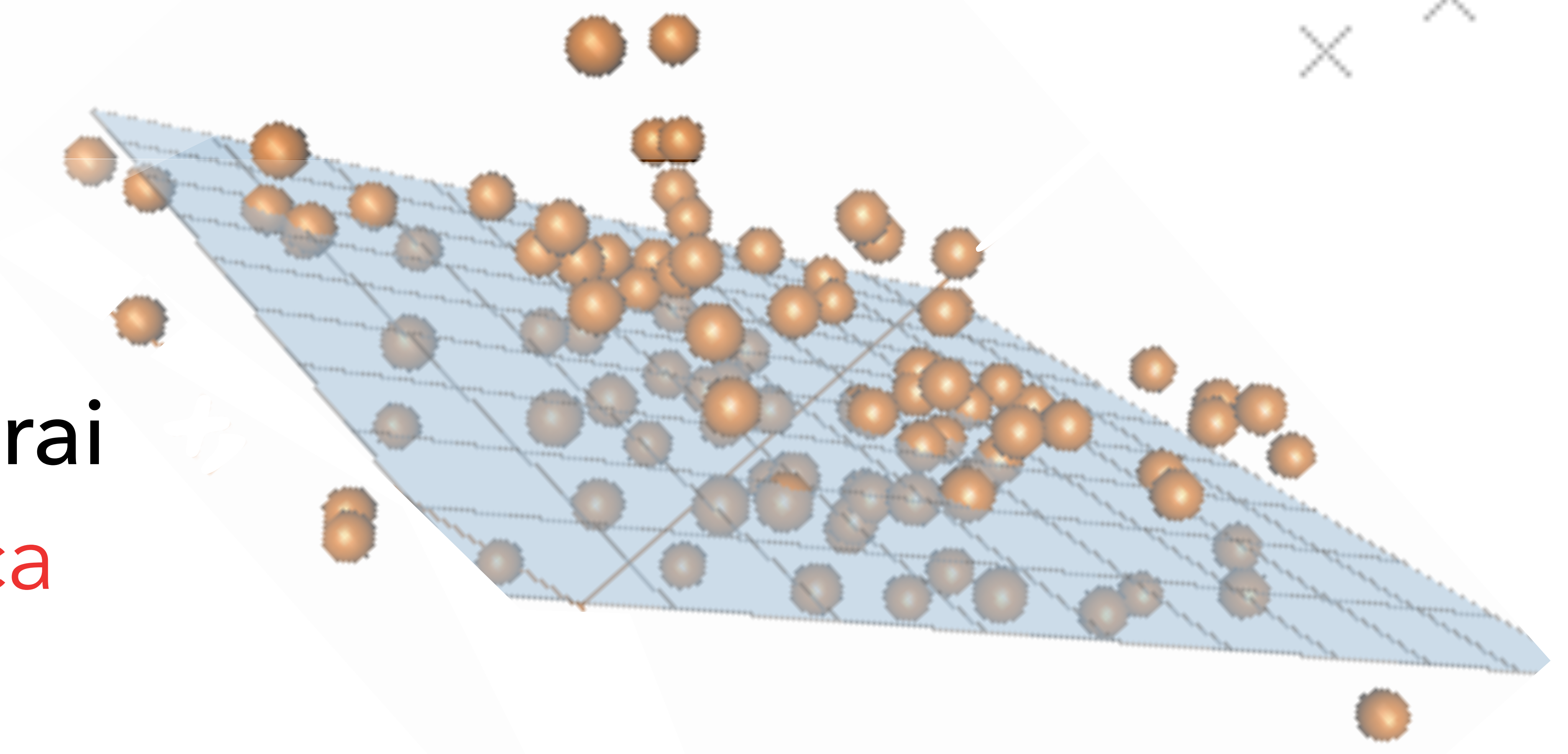
2c – Least-squares: QR & Regularization

Derek Nowrouzezahrai
derek@cim.mcgill.ca



Numerical Methods for WLS Solutions

We've formulated the LS solution of an overdetermined system as the solution of a simpler square system

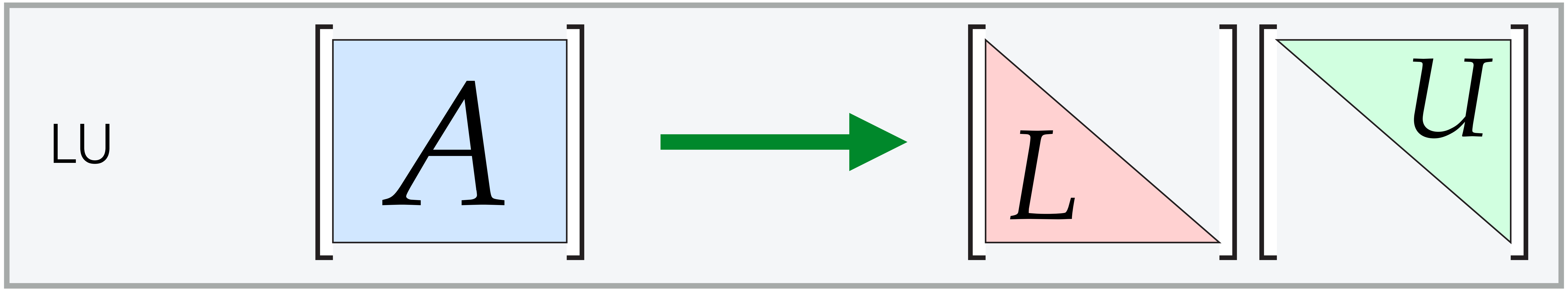
- this is an instance of a recurring, higher-level strategy:
  - decompose a difficult problem into easier problem(s)

$$\left[\mathbf{A}\right]\left[\mathbf{x}\right] = \left[\mathbf{b}\right] \longrightarrow \left[\mathbf{A}^T\mathbf{A}\right]\left[\mathbf{x}\right] = \left[\mathbf{A}^T\mathbf{b}\right]$$

Solving Linear Systems of Equations

Unsurprisingly, we've seen an example of this concept...

- the LU decomposition applies this principle



square system / fully-constrained → triangular systems

Numerical Methods for WLS Solutions

Given the *mathematical* solution of the square system formed by the normal equations, we can immediately try to solve the problem *numerically* with LU

$$\left[\mathbf{A}^T\mathbf{A}\right] = \left[L\right]\left[U\right]$$

$$\mathbf{x} = \left[U\right]^{-1}\left(\left[L\right]^{-1}\left[\mathbf{A}^T\mathbf{b}\right]\right)$$

- in fact, observe that $\mathbf{A}^T\mathbf{A}$ is *symmetric positive definite*, so …

Numerical Methods for WLS Solutions

Recall that the LU factorization of an SPD system is the Cholesky decomposition

- computing the Cholesky decomposition is faster than LU

$$\left[\mathbf{A}^T\mathbf{A}\right] = \left[L\right]\left[L^T\right]$$

$$\mathbf{x} = \left[L^T\right]^{-1}\left(\left[L\right]^{-1}\left[\mathbf{A}^T\mathbf{b}\right]\right)$$

Numerical Methods for WLS Solutions

While this is a good place to start, there are several reasons why doing so is not a good idea:

1. we can lose floating point precision **even before** solving – e.g., in the actual formation of the $\mathbf{A}^T\mathbf{A}$ matrix product

Consider $\underbrace{\begin{bmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{bmatrix}}_{\mathbf{A}}$ for $0 < \epsilon < \sqrt{\epsilon_m}$, then $\underbrace{\begin{bmatrix} 1+\epsilon^2 & 1 \\ 1 & 1+\epsilon^2 \end{bmatrix}}_{\mathbf{A}^T\mathbf{A}}$ is *numerically* singular

Numerical Methods for WLS Solutions

While this is a good place to start, there are several reasons why doing so is not a good idea:

2. and, even if we avoid floating point issues in the formation of the system, don't forget that the conditioning of $\mathbf{A}^T\mathbf{A}$ is worse than the condition of $\mathbf{A}$

$$\mathrm{cond}(\mathbf{A}^T\mathbf{A}) \approx \mathrm{cond}(\mathbf{A})^2$$

- all of this doesn't necessarily mean that using the normal equations is "bad", but rather that you must remain mindful of these conditions

Numerical Methods for WLS Solutions

For these reasons, it would be beneficial to work directly with the LS over-determined system (*i.e.*, to never form $\mathbf{A}^T\mathbf{A}$)

- can we apply Gaussian elimination directly to the over-determined matrix $\mathbf{A}$ to generalize the LU decomposition to tall matrices?
  - we'll see that the LU decomposition does not preserve the Euclidean norm, making it difficult to reason about (least-squares) distances
  - this is especially important in the least-squares setting, where we are interested in the utility of *a family of possible solutions*

Warping due to LU

Recall that the LU decomposition relies on composing fundamental matrix operations: *scaling* and *elimination* matrices

- these operations are all lower triangular (and so, too, their inverses)

  - scaling matrices: $\begin{bmatrix} \alpha & & \\ & \beta & \\ & & \gamma \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{A}_{11} & \alpha\mathbf{A}_{12} & \alpha\mathbf{A}_{13} \\ \beta\mathbf{A}_{21} & \beta\mathbf{A}_{22} & \beta\mathbf{A}_{23} \\ \gamma\mathbf{A}_{31} & \gamma\mathbf{A}_{32} & \gamma\mathbf{A}_{33} \end{bmatrix}$

  - elimination matrices: $\begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} + c\mathbf{A}_{11} & \mathbf{A}_{22} + c\mathbf{A}_{12} & \mathbf{A}_{23} + c\mathbf{A}_{13} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$

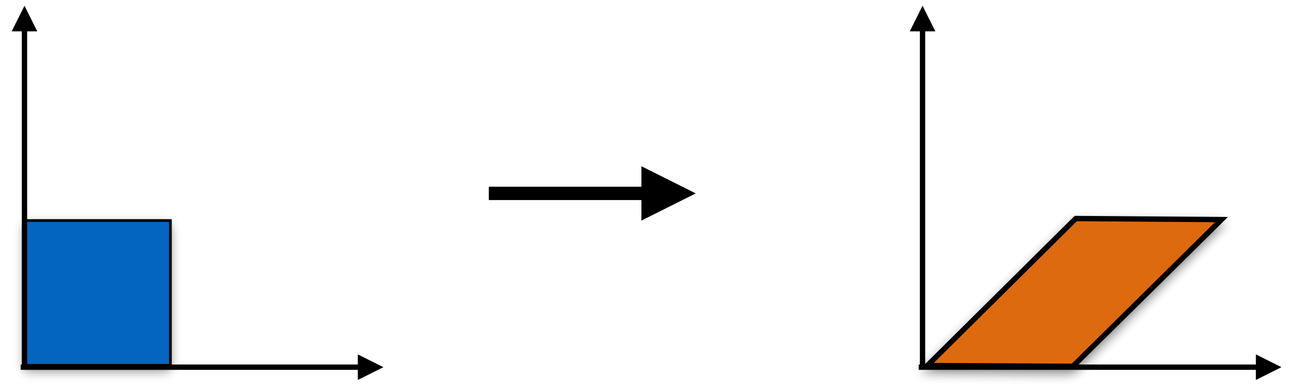
Warping due to LU

Consider the **geometric** perspective – *i.e.*, of the associated linear map – for these matrices, and its impact on *transformed distances*

- scaling matrices perform a <u>scale</u> on the space
- elimination matrices $\mathbf{E}_i$ <u>shear</u> the space

$$\begin{bmatrix} \mathbf{v}'_x \\ \mathbf{v}'_y \end{bmatrix} = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} + c\mathbf{A}_{11} & \mathbf{A}_{22} + c\mathbf{A}_{12} & \mathbf{A}_{23} + c\mathbf{A}_{13} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$



Warping due to LU

Distances in the untransformed coordinate system are not preserved after transformation:

$$\|\mathbf{Ex}\|_2 \neq \|\mathbf{x}\|_2$$

- what's more, the *composition* of many, *e.g.*, elimination matrices has a compounding effect on this warping of distances

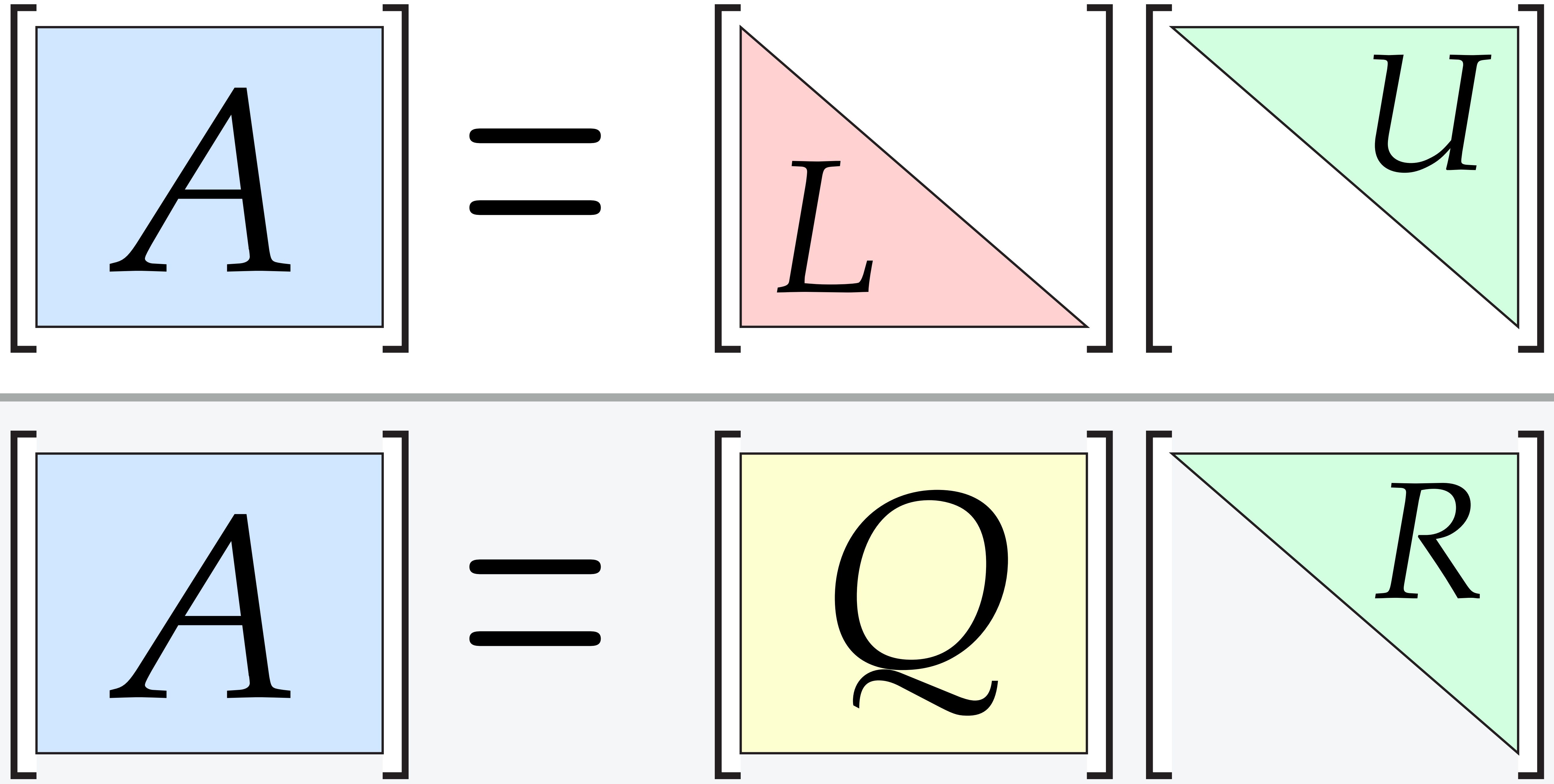
$$\mathbf{U} = (\mathbf{E}_n \ldots \mathbf{E}_1)\mathbf{A}$$

QR Decomposition

Solving Linear Least-squares Systems

LU

QR



Motivation – LU decomposition

LU is elegant in that it reduces the solution of an arbitrary square system to that of (two) simpler triangular systems

$$\mathbf{x} = \begin{bmatrix} U \end{bmatrix}^{-1} \left(\begin{bmatrix} L \end{bmatrix}^{-1} \mathbf{b} \right)$$

Motivation – LU decomposition

The way it does this is by incrementally "zero'ing out" column entries below the pivot

- unfortunately, the $\mathbf{E}_i$ shear space and do not preserve the Euclidean norm

Goal: reduce over-determined systems to a simpler triangular form

- $\mathbf{E}_i$'s are not a suitable building block for such a construction

pivot element

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

$$\mathbf{E}_7\mathbf{E}_6\mathbf{E}_5\mathbf{E}_4\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{A}$$

QR Decomposition – basic idea

We'll still "zero out" elements from a column but, now, using only transformations that preserve Euclidean norm

- what kind of transformations meet these needs?

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

$$\mathbf{A}$$

QR Decomposition – challenge

Simplified problem statement:

- zero out all the elements in a vector $\mathbf{x}$, except its first element
- do so while preserving the Euclidean norm

$$\mathbf{Q}\mathbf{x} = \mathbf{Q}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{x}}_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \|\mathbf{Q}\mathbf{x}\|_2 = \|\mathbf{x}\|_2$$

QR Decomposition – challenge

Consider zero'ing out all of the entries in the first column:

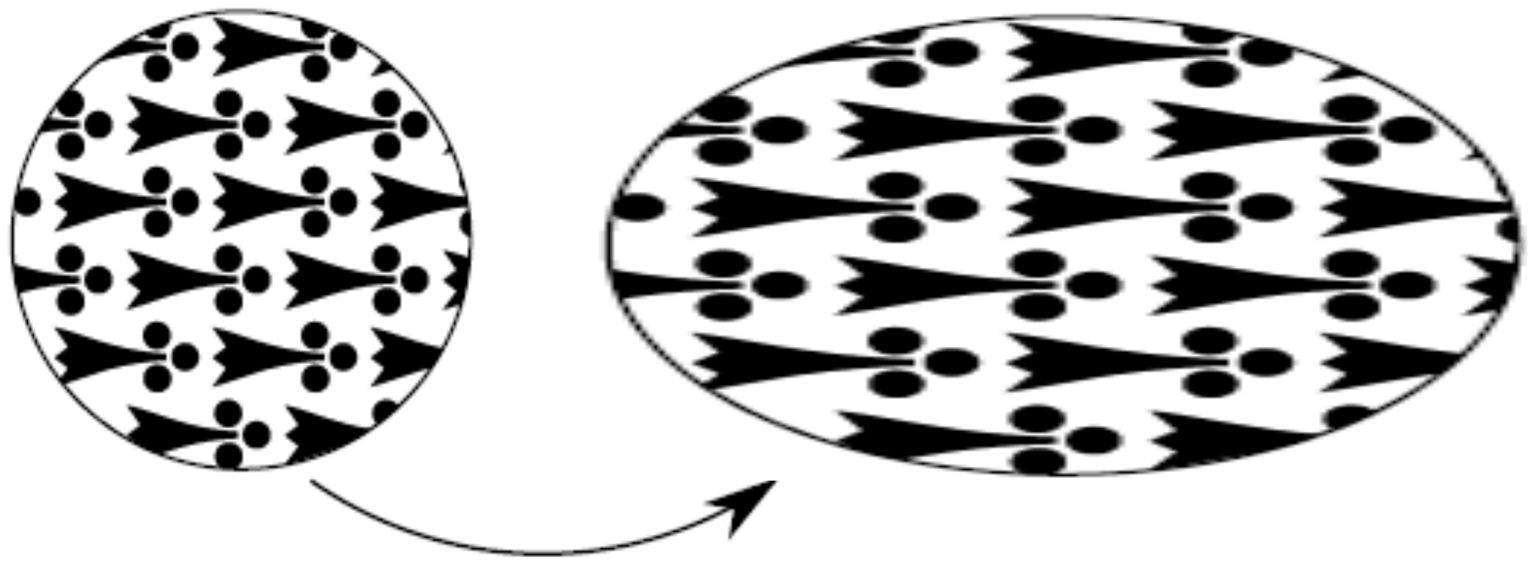
- geometrically, the column $\mathbf{a}_1$ is an arbitrary vector in $\mathbf{R}^n$
- we seek a transformation (or sequence of transformations) that "zero out" all but one of its elements
  - thoughts? options?

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

Restriction to Isometric Transformations

If we're limited to only using transformations that preserve Euclidean norm, what are our options?

- scales?
    - No! Do not preserve Euclidean norm:



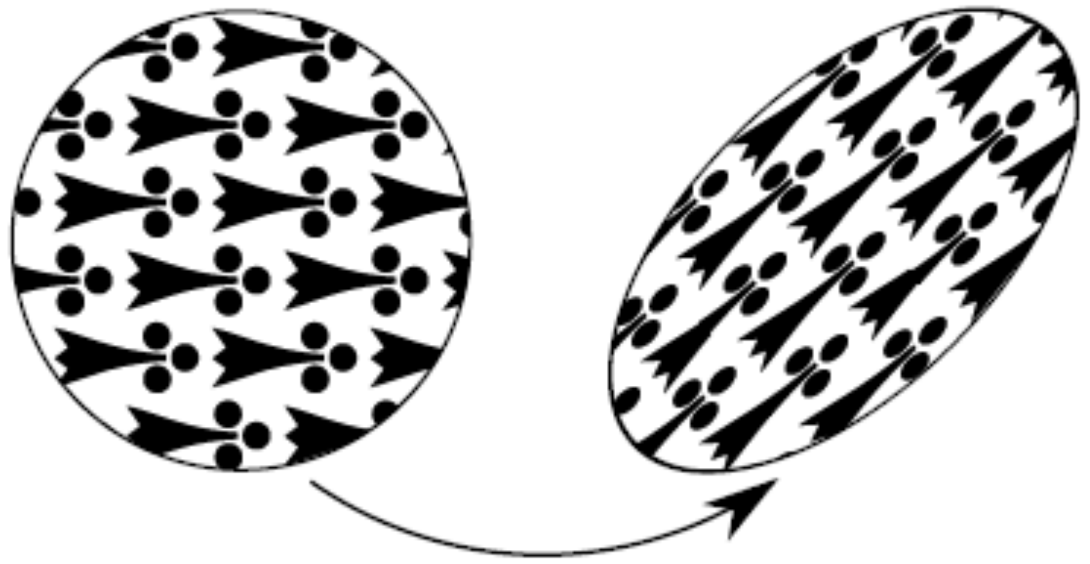
Not isometric

Restriction to Isometric Transformations

If we're limited to only using transformations that preserve Euclidean norm, what are our options?

- shears?

  - No! Do not preserve Euclidean norm:

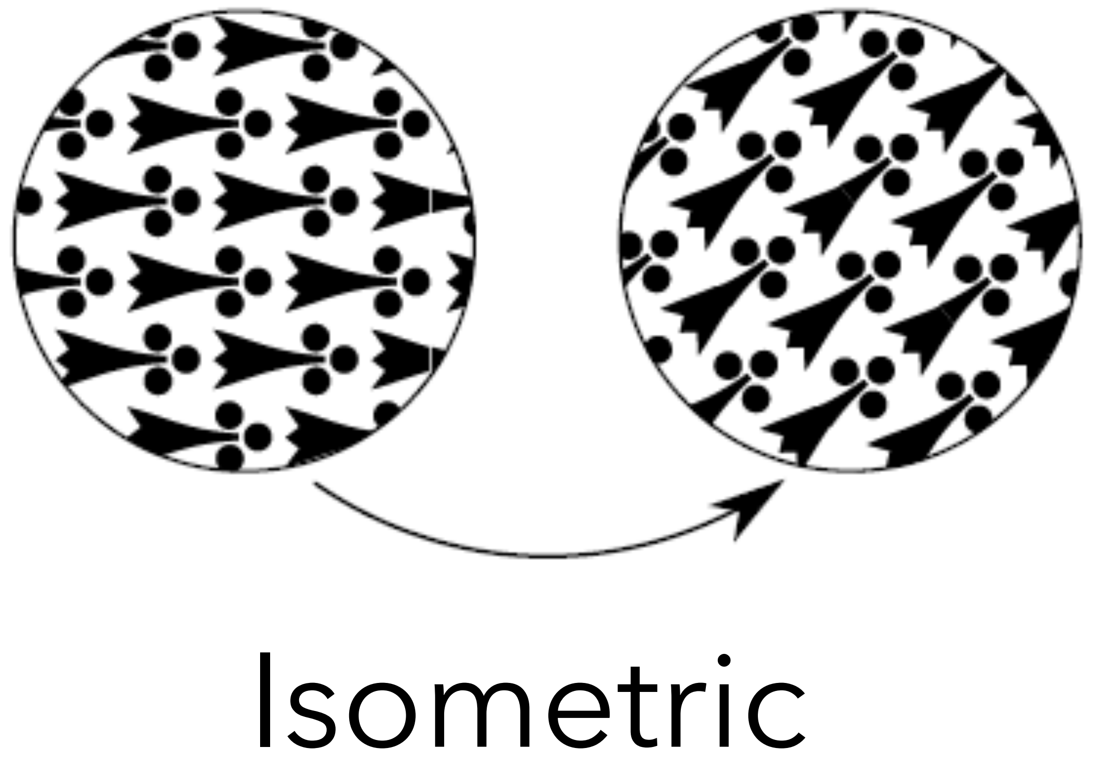


Not isometric

Restriction to Isometric Transformations

If we're limited to only using transformations that preserve Euclidean norm, what are our options?

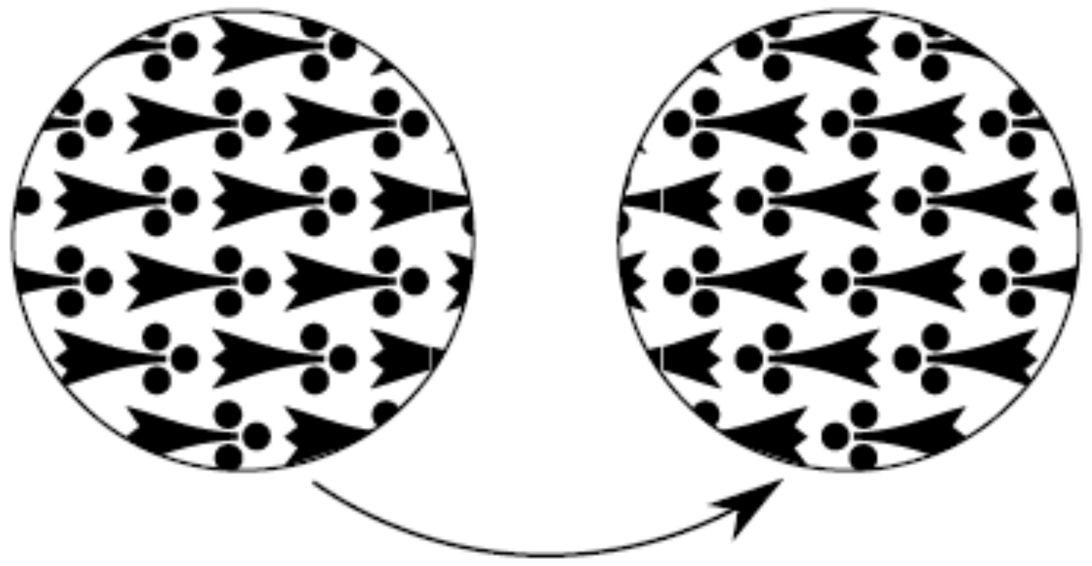
- rotations?
    - Yes!



Restriction to Isometric Transformations

If we're limited to only using transformations that preserve Euclidean norm, what are our options?

- reflections?
  - Yes!



QR Decomposition – challenge

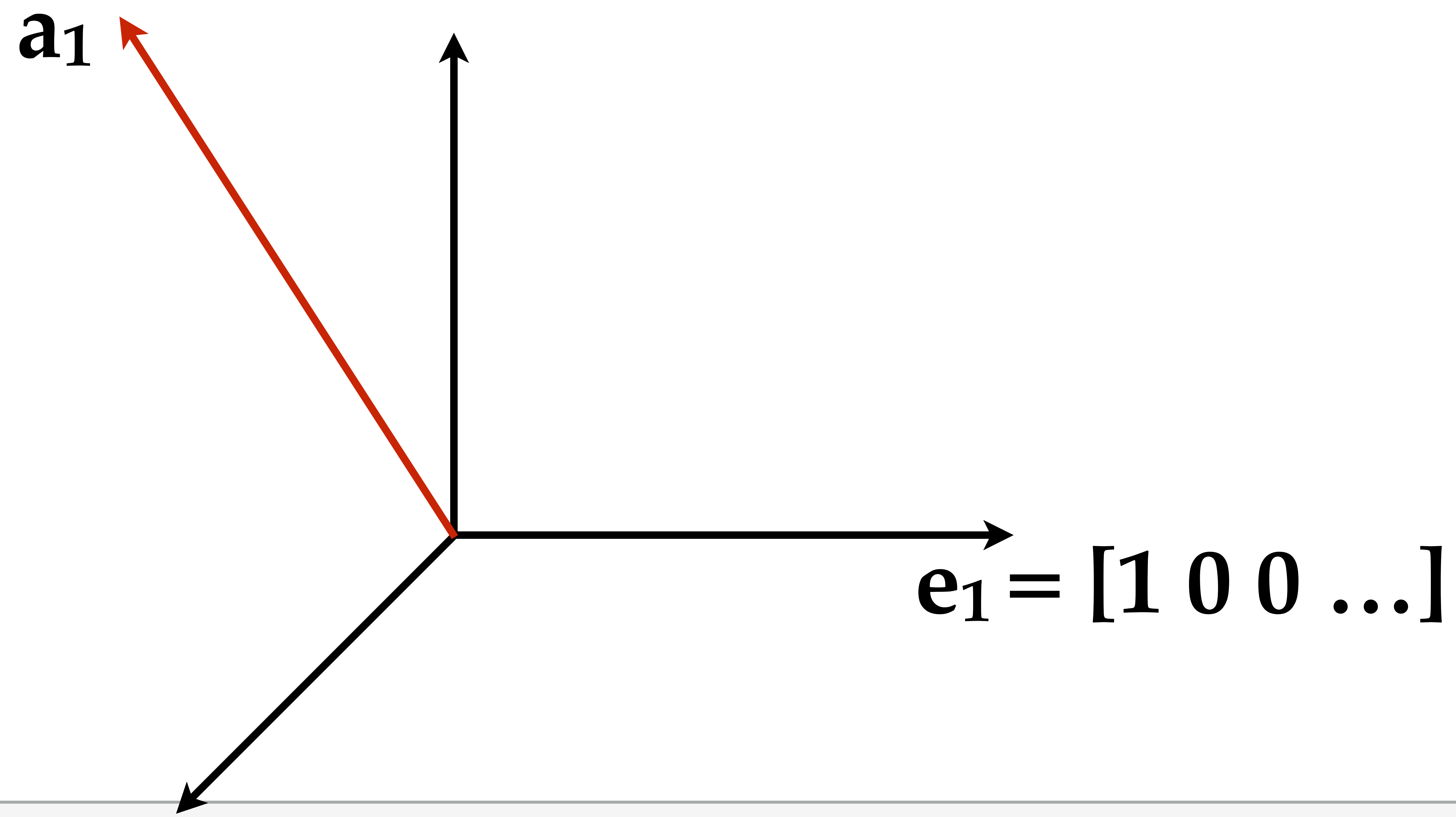
Consider zero'ing out all of the entries in the first column:

- geometrically, the column $\mathbf{a}_1$ is an arbitrary vector in $\mathbf{R}^n$
- we seek a transformation (or sequence of transformations) that "zero out" all but one of its elements
  - plenty of options… thoughts?
    - rotations

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

Rotation to desired axis

For the first column $\mathbf{a}_1$, one way to zero out all but one of the elements is to rotate the column vector to align with the first canonical basis vector $\mathbf{e}_1$

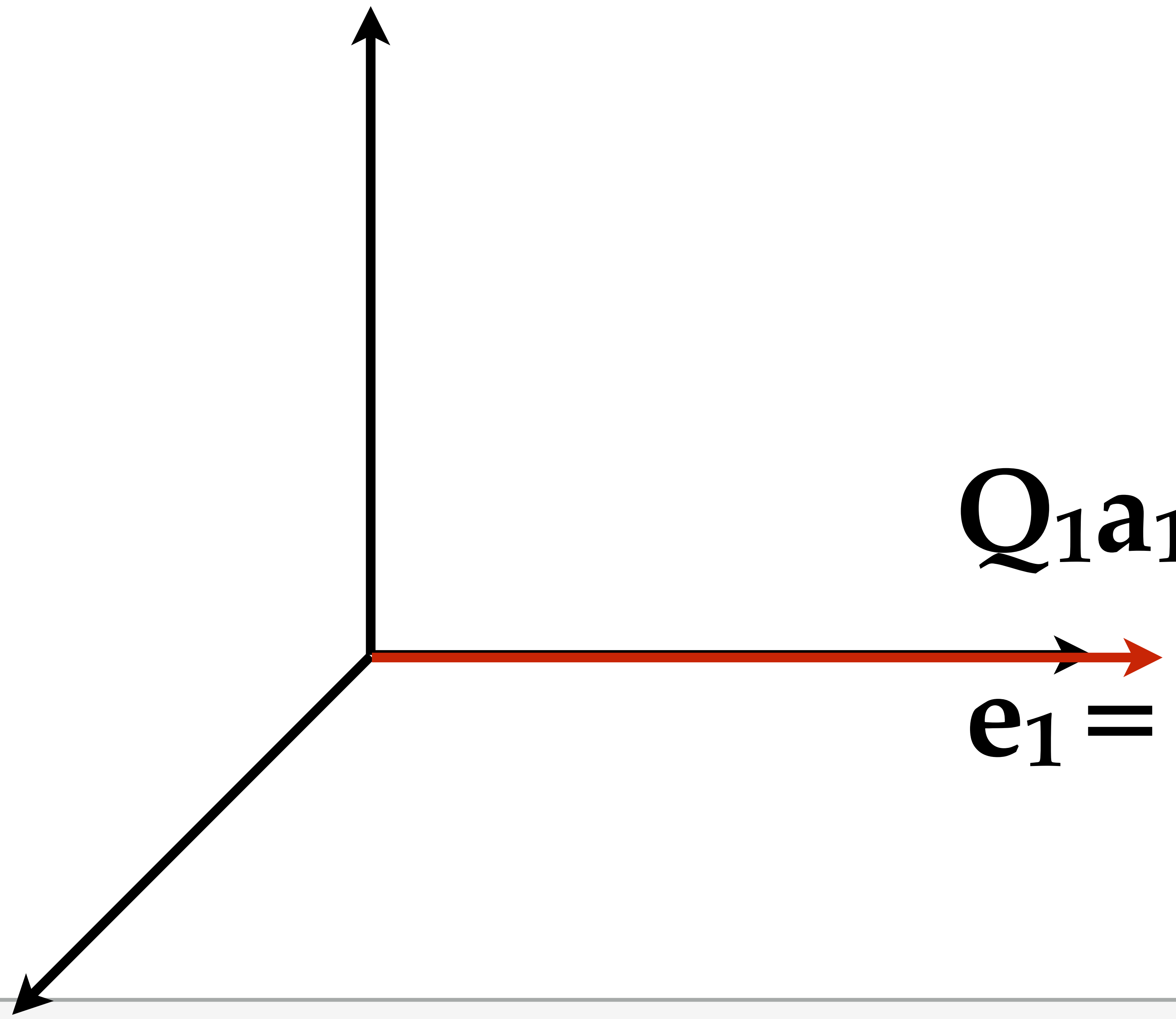


$$\begin{array}{cccc} \mathbf{a}_1 & & & \\ \end{array}$$
$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

Rotation to desired axis

We know how to come up with this rotation matrix $\mathbf{Q}_1$

- change of basis: the inverse of the rotation $\mathbf{R}$ from $\mathbf{e}_1$ to $\mathbf{a}_1$
- inverse a rotation $\mathbf{R}$ is its transpose: $\mathbf{R}^{-1} = \mathbf{R}^T$



$\mathbf{Q}_1\mathbf{a}_1 = [x\ 0\ 0\ \ldots]$

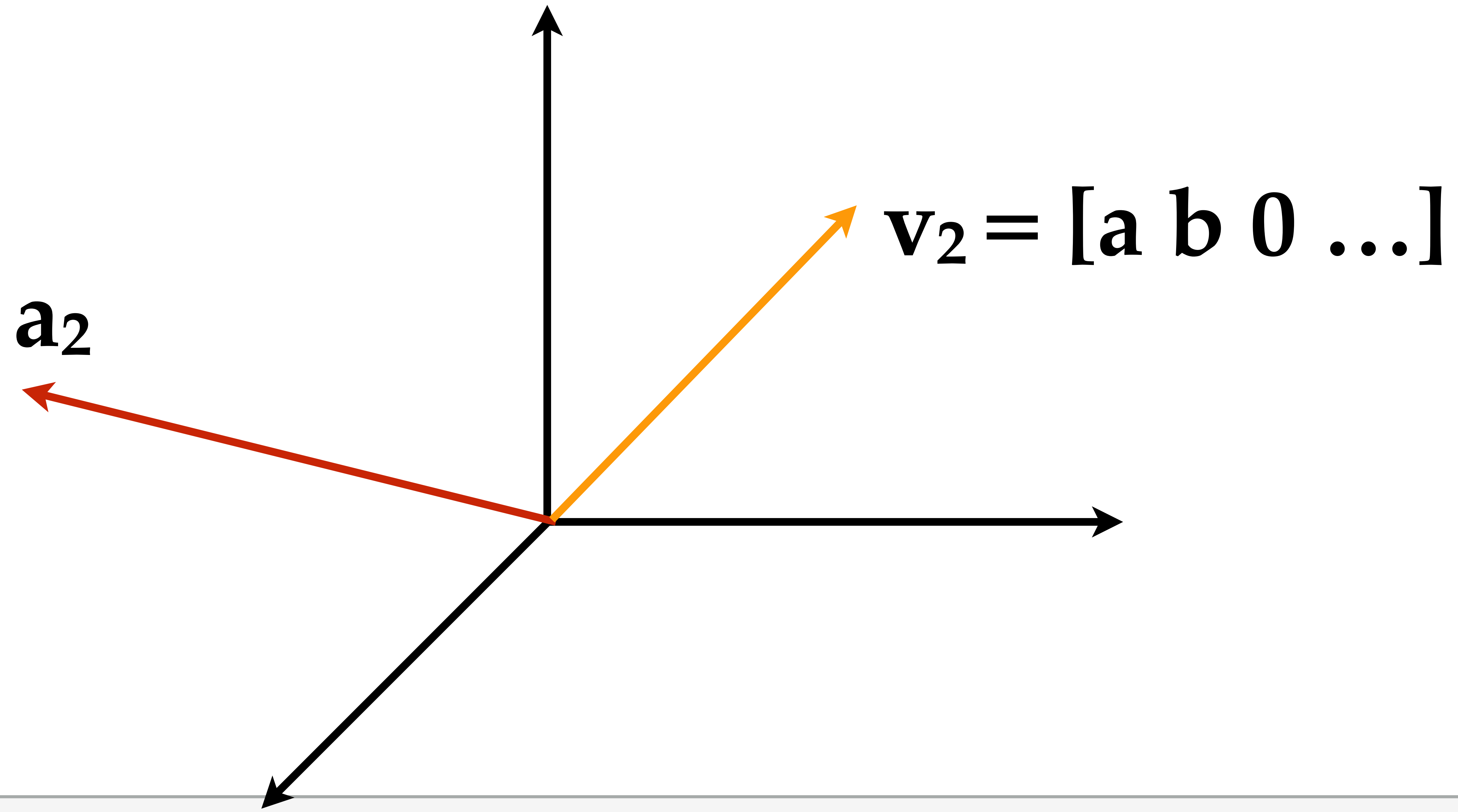
$\mathbf{e}_1 = [1\ 0\ 0\ \ldots]$

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

Rotation for other columns

Beyond the first column, things can get a little trickier…

- not rotating onto a canonical basis vector $\mathbf{e}_i$ anymore

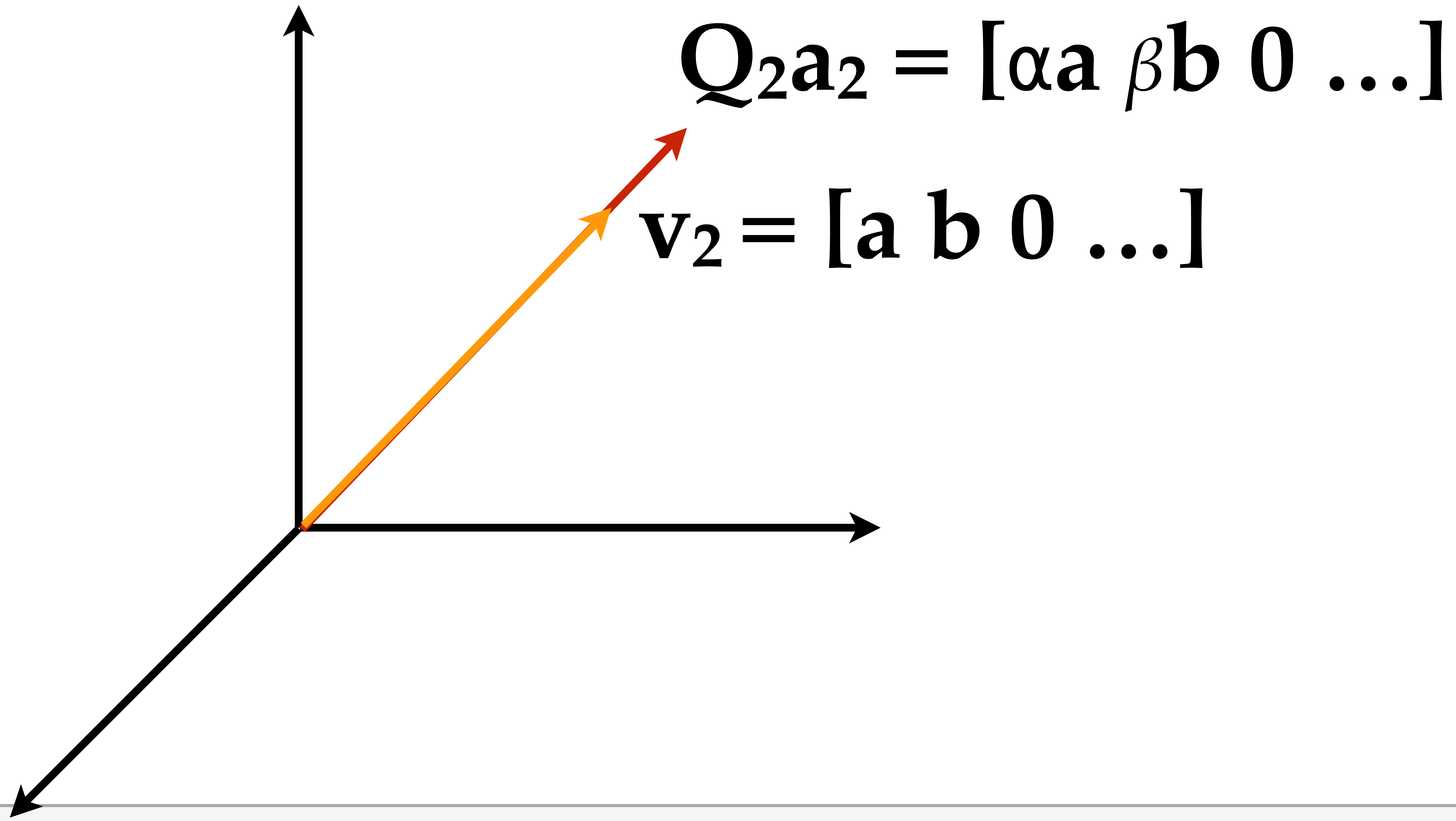


$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

Rotation for other columns

Beyond the first column, things can get a little trickier…

- not rotating onto a canonical basis vector $\mathbf{e}_i$ anymore
- can still formulate this as a rotation… ideas?



$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Isometric Transformations

Consider zero'ing out all of the entries in the first column:

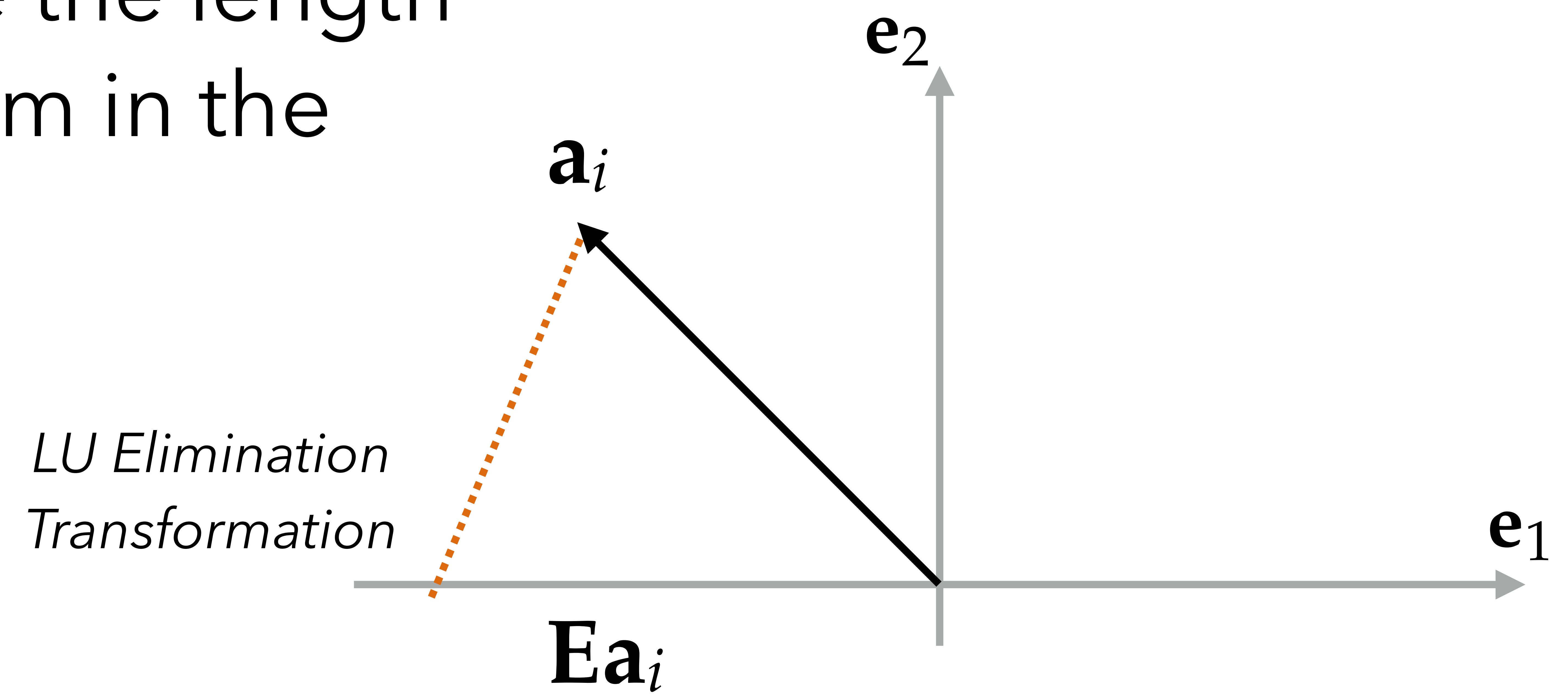
- geometrically, the column $\mathbf{a}_1$ is an arbitrary vector in $\mathbf{R}^n$
- we seek a transformation (or sequence of transformations) that "zero out" all but one of its elements
  - plenty of options… thoughts?
    - rotations
    - reflections

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

Householder Reflections

Visually we want to align, e.g., $\mathbf{a}_1$ with a canonical axis $\mathbf{e}_i$

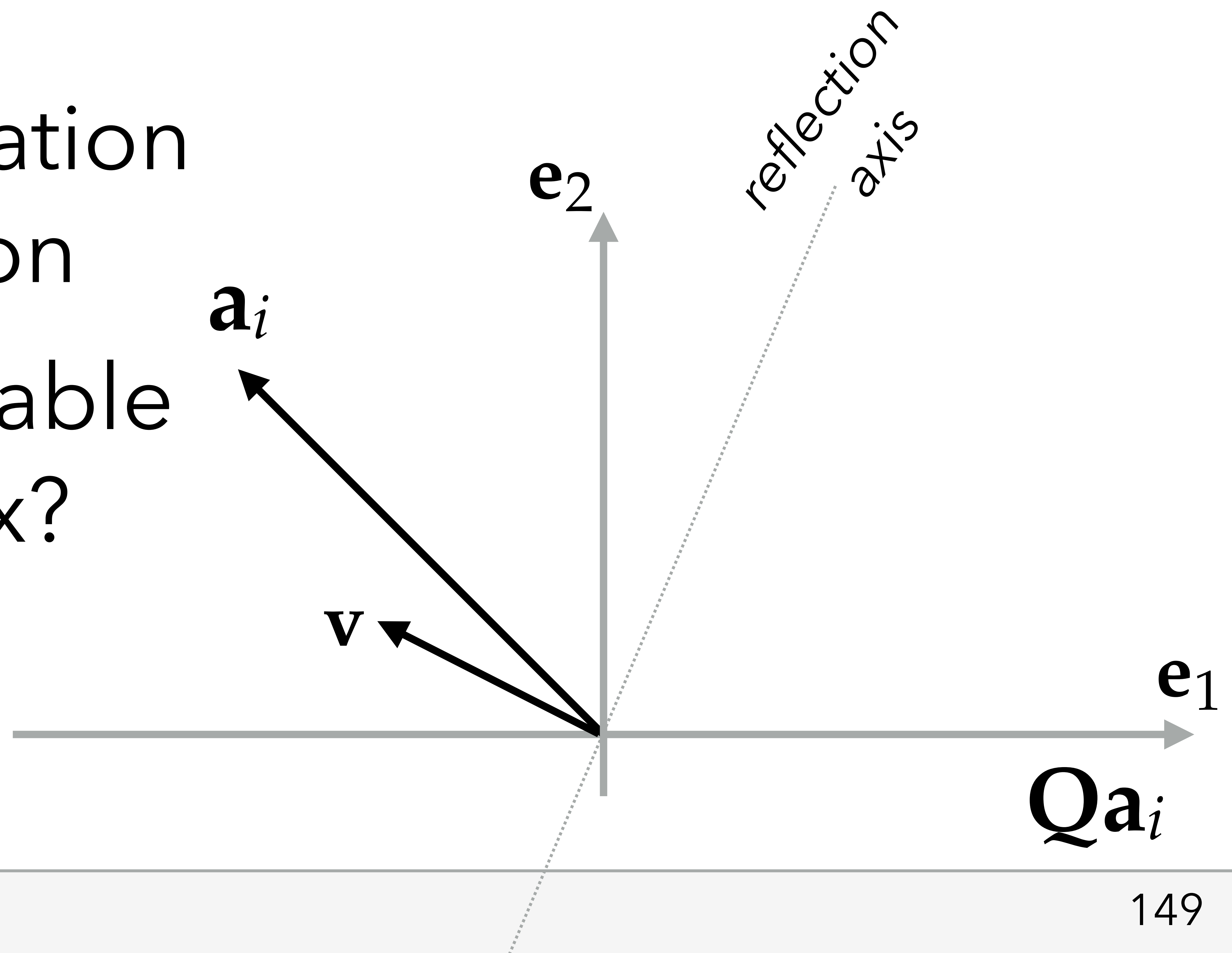
- the LU elimination transformations do so by shearing the vector onto the canonical axis
  - this **does not** preserve the length of $\mathbf{a}_1$, which is a problem in the least squares setting



Householder Reflections

Visually we want to align, e.g., $\mathbf{a}_1$ with a canonical axis $\mathbf{e}_i$

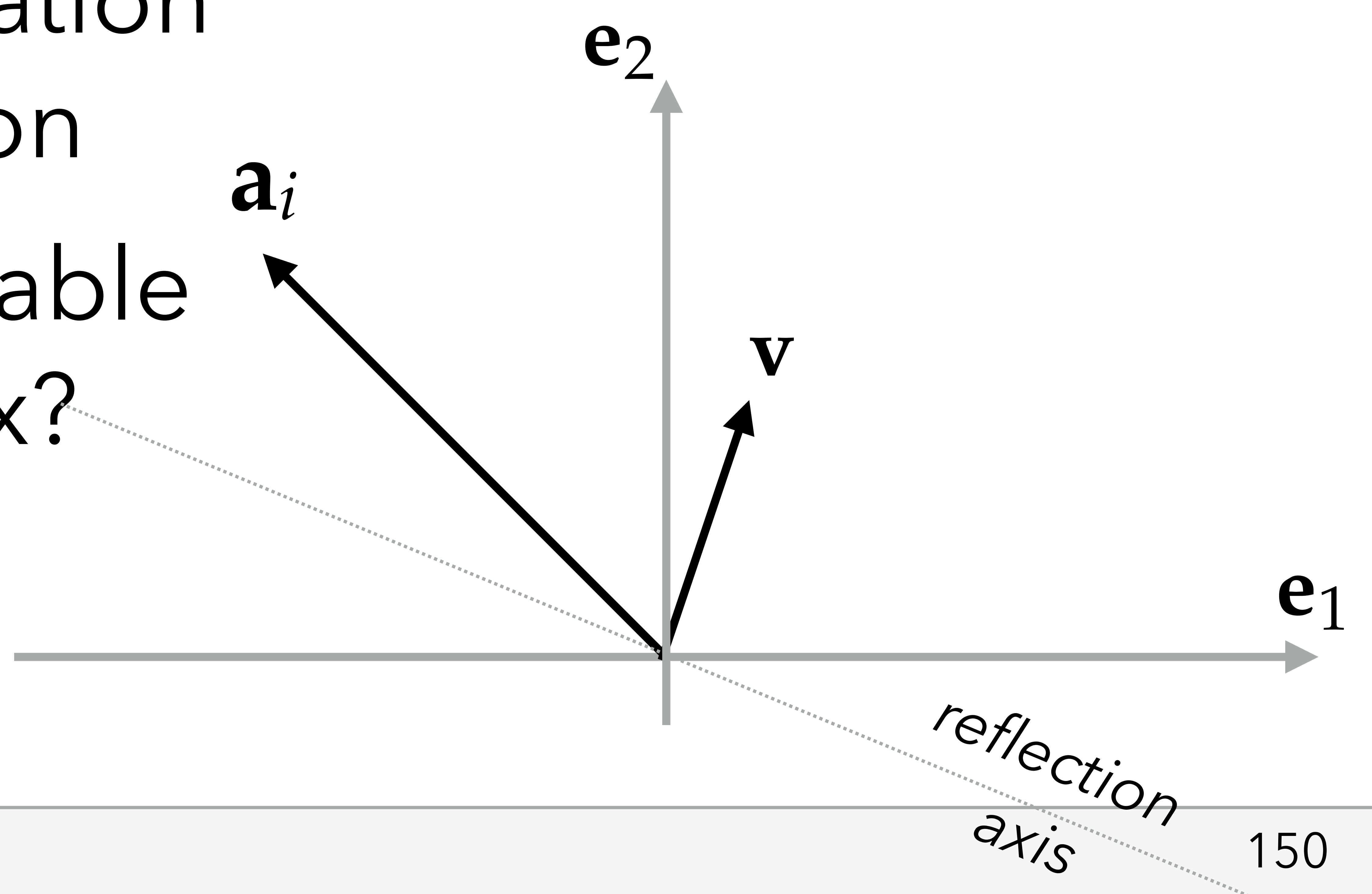
- the two alternatives we outlined for transforming $\mathbf{a}_j$ onto $\mathbf{e}_i$ are rotations and reflections
  - we can construct a suitable rotation matrix as an ONB transformation
  - but how do we construct a suitable reflection transformation matrix?
    - is there a unique solution?



Householder Reflections

Visually we want to align, e.g., $\mathbf{a}_1$ with a canonical axis $\mathbf{e}_i$

- the two alternatives we outlined for transforming $\mathbf{a}_j$ onto $\mathbf{e}_i$ are rotations and reflections
  - we can construct a suitable rotation matrix as an ONB transformation
  - but how do we construct a suitable reflection transformation matrix?
    - is there a unique solution?



Householder Reflections

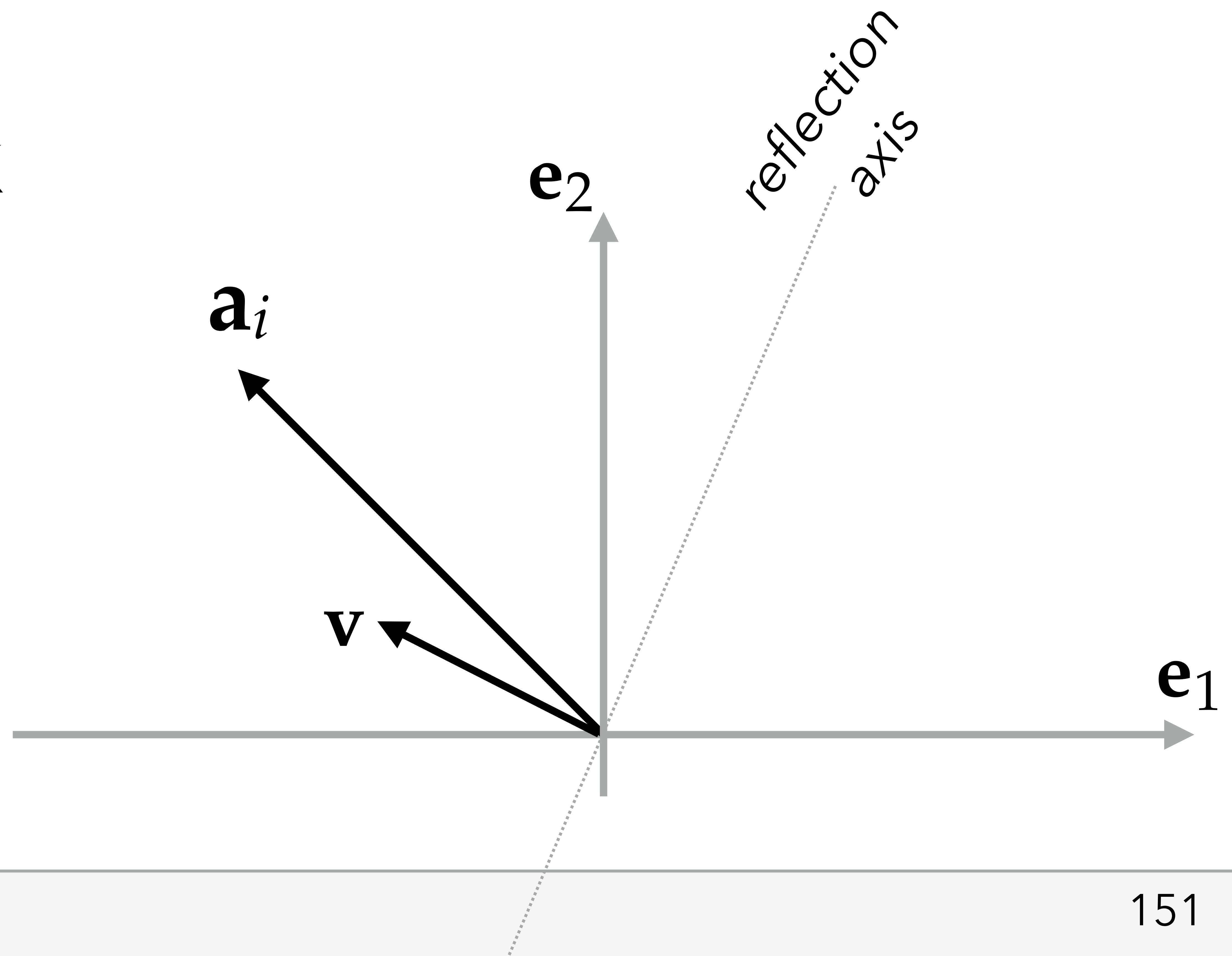
We obtain an expression relating $\mathbf{a}_j$ to its reflection about a reflection axis with unit normal $\mathbf{v}$ as:

$$\mathbf{a}_j - 2(\mathbf{v}^T \mathbf{a}_j)\ \mathbf{v}$$

The associated reflection matrix

$$\mathbf{Q} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}$$

- we omit the details behind choosing an appropriate $\mathbf{v}$



Recall – Orthogonal Transformations

Remember – whether rotations and/or reflections – isometric transformations are matrices of the form

$$\mathbf{Q} = \begin{pmatrix} | & & | \\ \mathbf{Q}_1 & \cdots & \mathbf{Q}_n \\ | & & | \end{pmatrix}$$

where the $\mathbf{Q}_i$ are mutually orthonormal

$$\mathbf{Q}^T\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1^T\mathbf{Q}_1 & \cdots & \mathbf{Q}_1^T\mathbf{Q}_n \\ \vdots & & \vdots \\ \mathbf{Q}_n^T\mathbf{Q}_1 & \cdots & \mathbf{Q}_n^T\mathbf{Q}_n \end{pmatrix} = \mathbf{I}$$

- preserve Euclidean distances
- preserve angles
- does not amplify errors in the least squares setting

QR Decomposition

Householder transformation

- excellent numerical properties (👍), hard to parallelize (👎)

Givens rotations

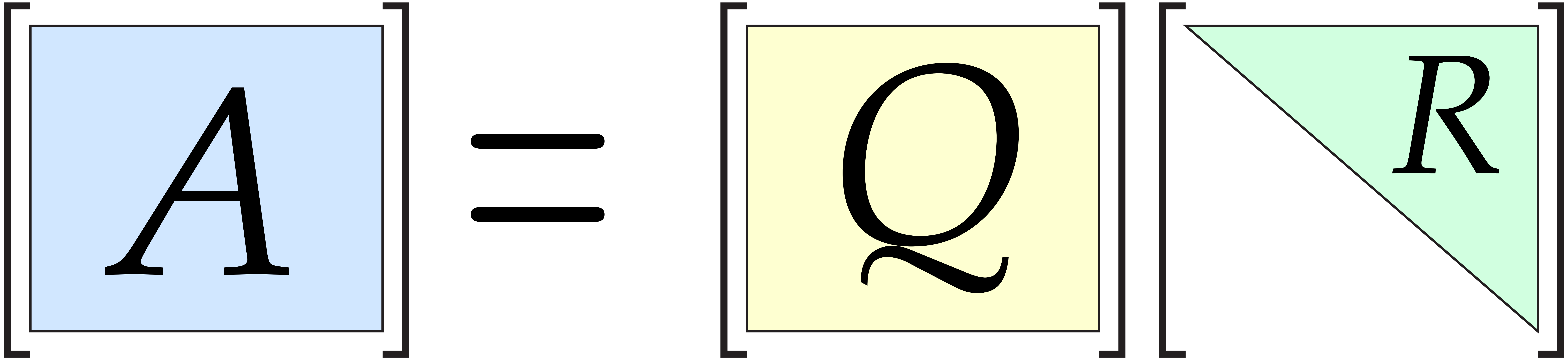
- excellent numerical properties (👍), parallelizable(👍), good for sparse problems (👍)

Gram-Schmidt orthogonalization

- simple to implement (👍), numerically unstable (👎)

QR Decomposition – square matrix

After composing the Q_i matrices into $Q^T = Q_n\, Q_{n\text{-}1} \ldots Q_1$ a fully-constrained (i.e., square) system matrix A can be expressed as the product of the (inverse/transpose) Q of and the resulting upper-triangular system R



QR Decomposition – overdetermined

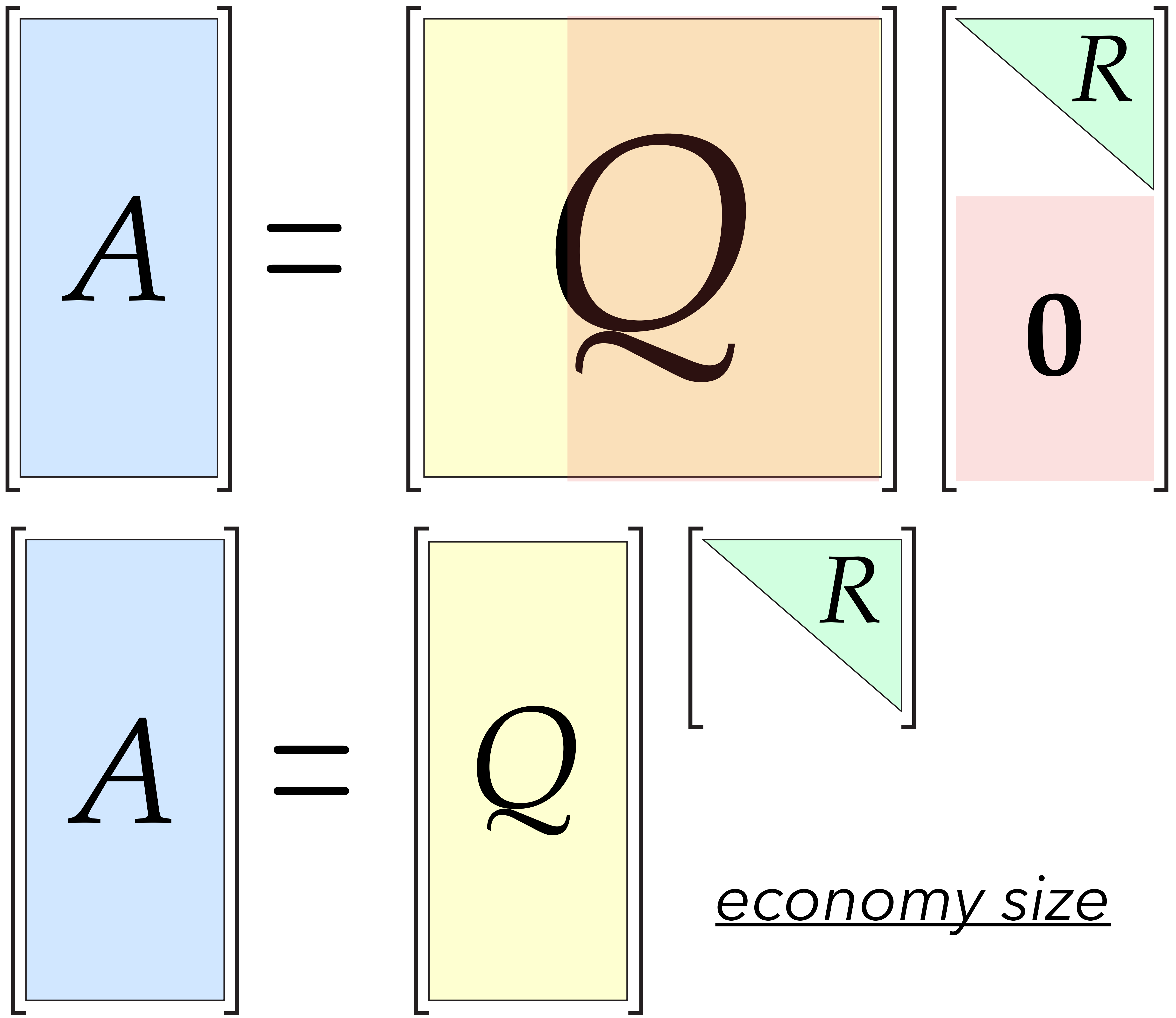
We can similarly apply QR to tall/overdetermined systems, where we arrive at a decomposition of the form:

$$\begin{bmatrix} A \end{bmatrix} = \begin{bmatrix} Q \end{bmatrix} \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix}$$

QR Decomposition – *"economy size"*

In overdetermined systems, the upper-triangular matrix R is zero-padded

- that means that a good deal of $\mathbf{Q}$ ends up being multiplied by 0s
- no need to store any of these "useless" components



QR – Solving Least Squares Systems

We write an expression for the least-squares solution as

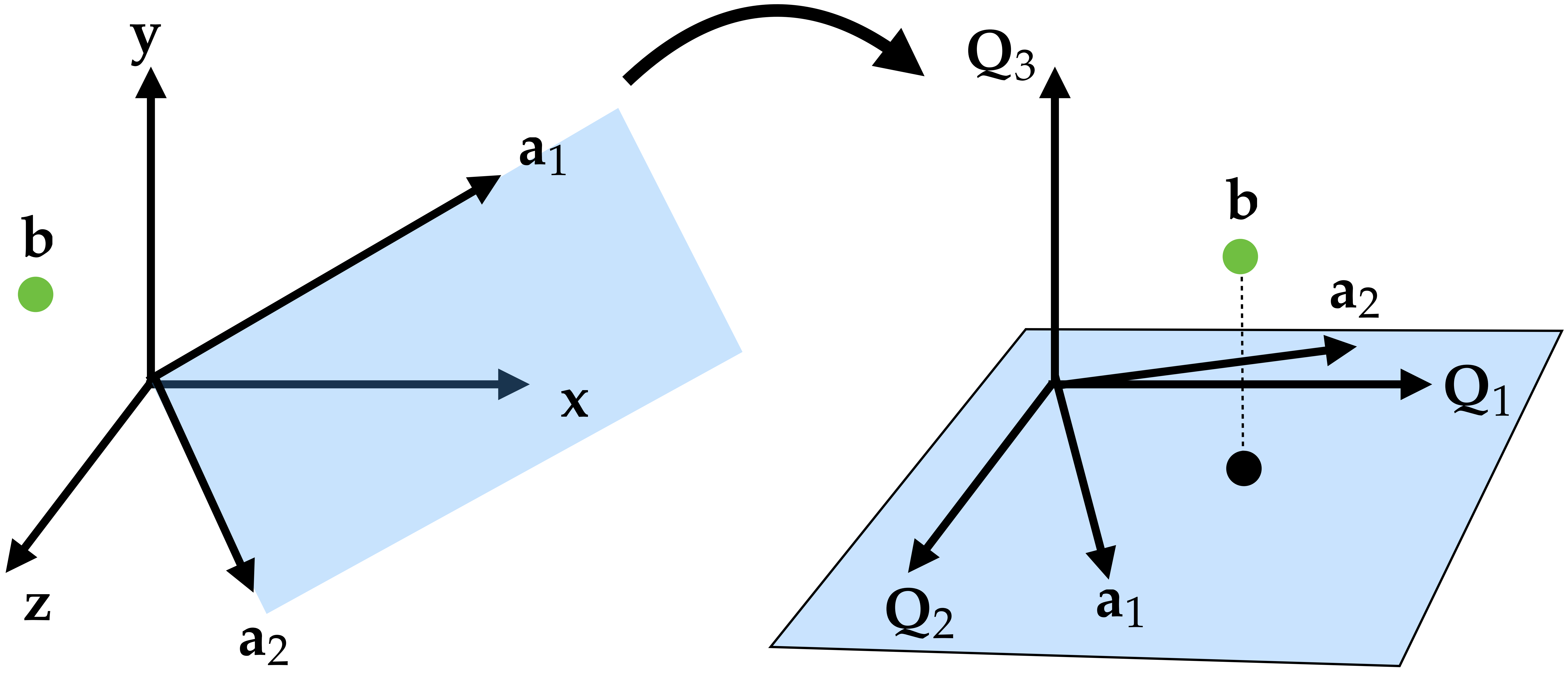
$$\mathbf{x} = \left[\mathbf{R}\right]^{-1}\left(\left[\boxed{\mathbf{Q}^T}\right]\mathbf{b}\right)$$

but be mindful that we don't have to actually invert $\mathbf{R}$

- instead, solve the upper-triangular system $\mathbf{R}\mathbf{x} = \mathbf{Q}^T\mathbf{b}$ using backward substitution!

QR – Solving Least Squares Systems

Geometrically, the decomposition transforms space so that a projection* of $\mathbf{b}$ yields the least-squares solution



Supplemental Readings

[Ascher & Greif] Section 6.1 – 6.3

Least squares and the normal equations, orthogonal transformations and QR, Householder and Gram-Schmidt

[Heath] Sections 3.1 – 3.5

Linear least squares, existence and uniqueness, problem transformations, orthogonalization methods

[Solomon] Chapter 4

Column spaces, QR

Regularization

Deconvolution example – re-visited

(what we expect)

(what actually happens)

[Jason Cole]

Understand why the world explodes…

Why? Conditioning!

In linear systems, the condition number of a system's matrix measures how sensitive a solution is to small deviations in input

Why not compute $\mathrm{cond}(\mathbf{A})$ before proceeding with a *solve*?

- *chicken & egg problem:* we want to know the condition number to find out if inverting the matrix is even possible, but to do so we need to …
- also, inverting the matrix is expensive

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

More efficient techniques exist…

```
>>> A = np.array(…)
>>> np.linalg.cond(A, p=2)
23.14
```

Regularization

Regularization is a general term associated with a process that is designed to improve the conditioning of an underlying system

- there are many different ways you can *regularize* a problem
- each approach has distinct trade-offs
    - there's no free lunch: in exchange for improved conditioning, a regularized solution will suffer from some balancing effect
        - e.g., decreased accuracy, decreased performance, bias, …

Tikhonov Regularization

Tikhonov regularization is perhaps the most common regularization method for linear systems

- simple premise: assuming that a symptom of an unstable solution is that the solution $\mathbf{x}$'s squared magnitude grows to be too large, we can penalize solutions with large L_2 norms

$$\mathbf{x}_{\text{solution}} = \operatorname{argmin}\left[\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2\right]$$

- here, the regularization factor λ is a user parameter that controls the degree to which we prefer smaller solutions

Implementing Tikhonov Regularization

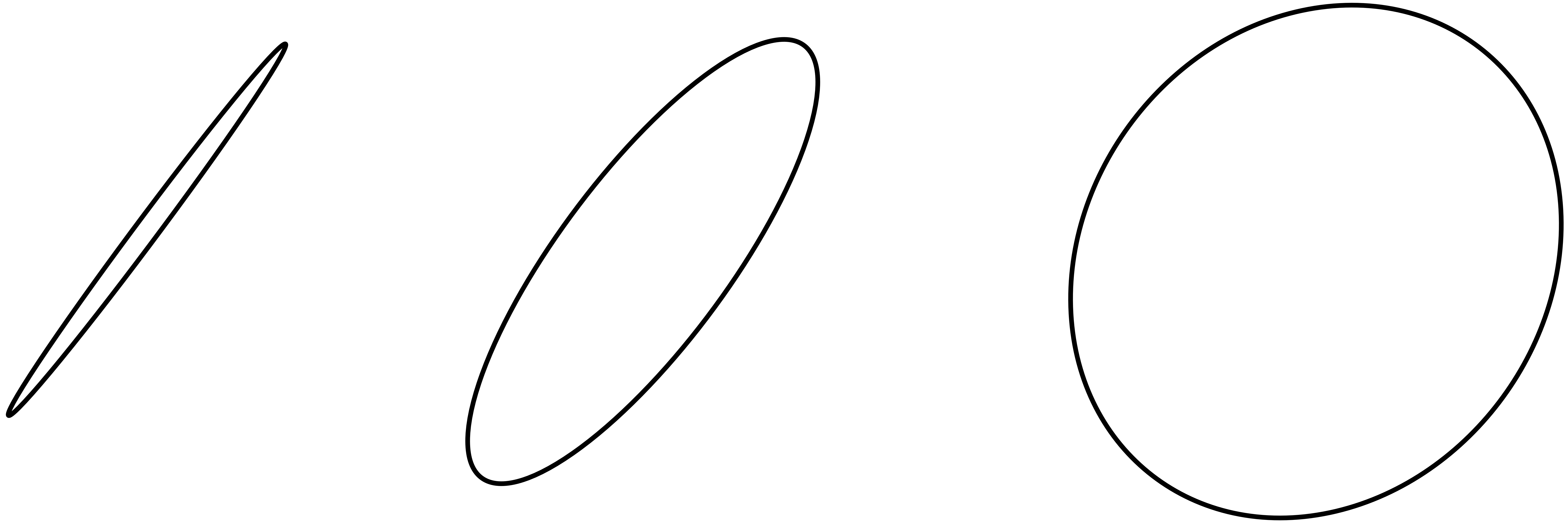
We can write the solution to any fully- or over-constrained linear system with the additional regularization term as

$$
\begin{aligned}
&\operatorname{argmin}\left[\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{b}^T\mathbf{A}\mathbf{x} + \|\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2\right] \\
= \;&\operatorname{argmin}\left[\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{b}^T\mathbf{A}\mathbf{x} + \|\mathbf{b}\|_2^2 + \lambda\mathbf{x}^T\mathbf{I}^T\mathbf{I}\mathbf{x}\right]
\end{aligned}
$$

- Tikhonov regularization can be easily integrated into both:
  - normal equations: add $\sqrt{\lambda}\mathbf{I}$ to $\mathbf{A}^T\mathbf{A}$ before solving, or
  - QR: compute the decomposition of $\begin{bmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{bmatrix}$ instead of $\mathbf{A}$

Regularization – Matrix Norm

In a poorly conditioned system, we can conceptually visualize the impact that Tikhonov regularization has on the matrix norm's space warping:



small λ medium λ large λ

Other regularizations

Tikhonov regularization is but one of many possible regularizers of a system

- also referred to as L_2-norm regularization, weight decay, magnitude penalization, etc.

Other general options include, but are not limited to:

- L_1-norm regularization: typically used to promote sparsity
- augmented system regularization