Lecture 7

The Eigen- & Singular Value Decompositions

Nonlinear matrix algorithms

CS328 - Numerical Methods for Visual Computing and Machine Learning

Prof. Wenzel Jakob

EPFL RGL Realistic Graphics Lab

MATH-111 recap

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

MATH-111 recap

Eigenvalue Eigenvector

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

MATH-111 recap

eigen, *noun (ger.)*: self

Eigenvalue → λ

Eigenvector → $\mathbf{v}$

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

MATH-111 recap

eigen, *noun (ger.)*: self

Eigenvalue → λ Eigenvector → $\mathbf{v}$

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

where

$$\mathbf{A} \in \mathbb{R}^{n\times n}, \;\; \lambda \in \mathbb{R}$$
$$\mathbf{v} \in \mathbb{R}^{n}, \|\mathbf{v}\| = 1$$

MATH-111 recap

eigen, *noun (ger.)*: self → Eigenvalue   Eigenvector

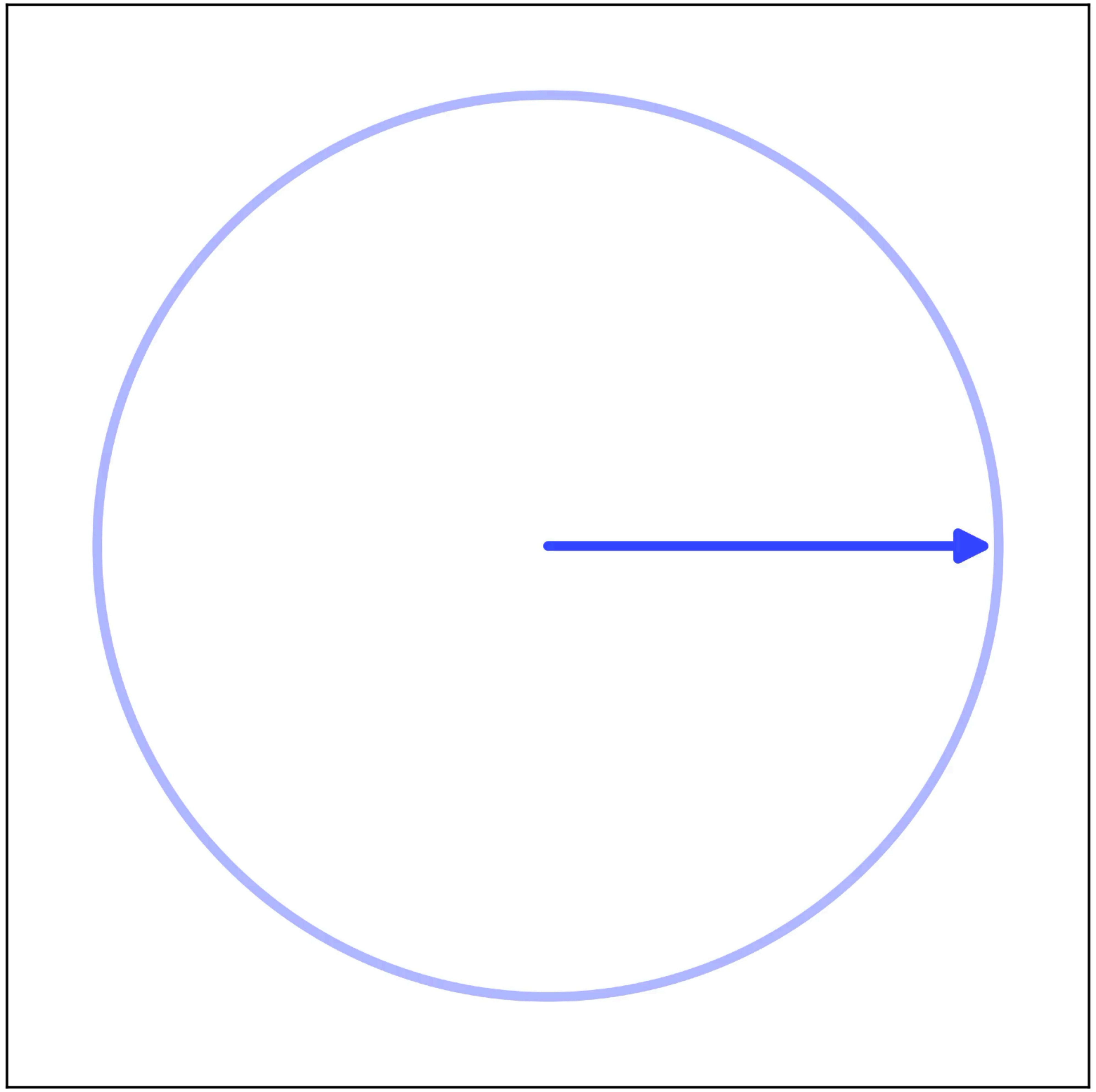
$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

where

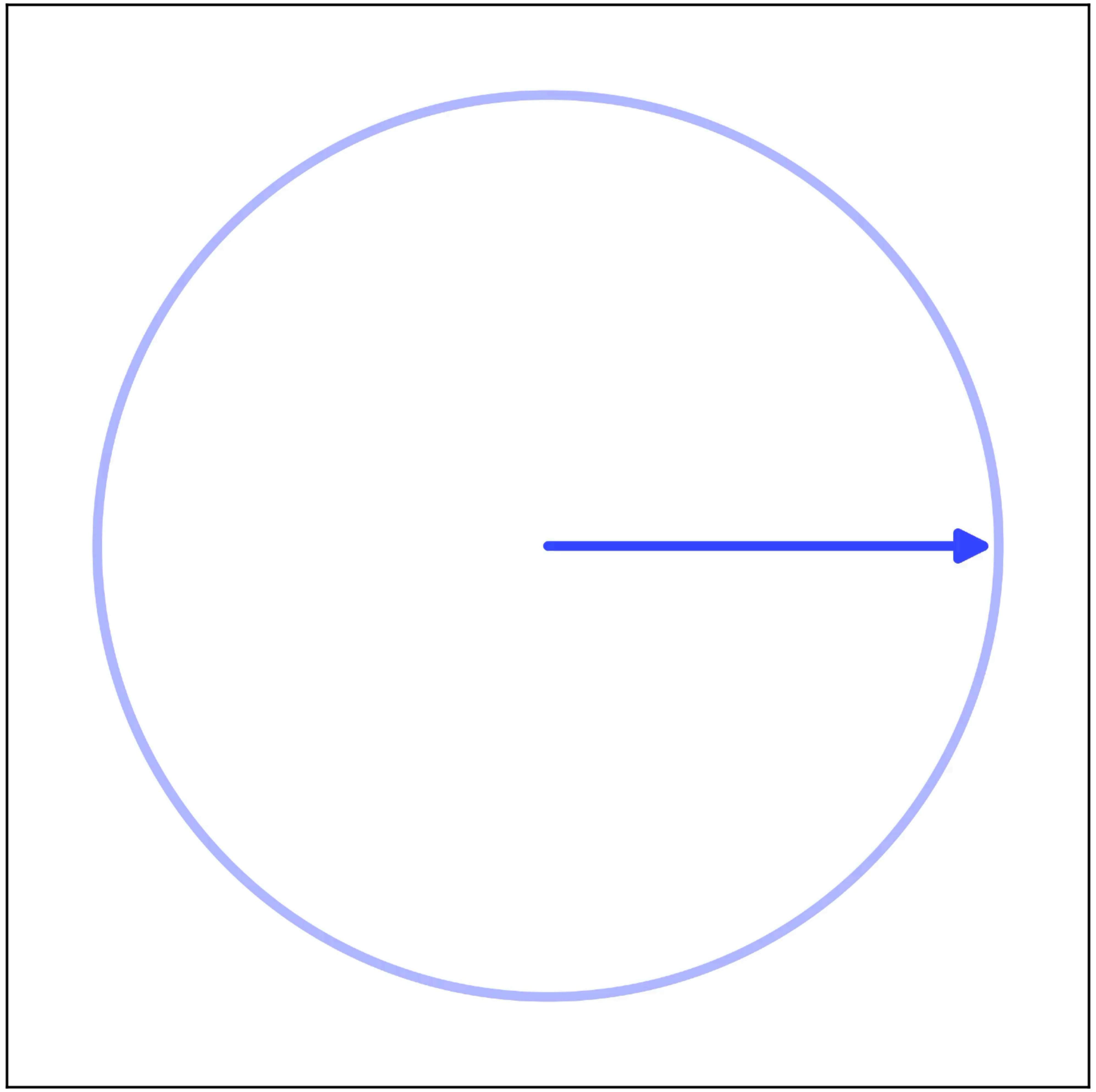
$$\mathbf{A} \in \mathbb{R}^{n\times n}, \ \ \lambda_i \in \mathbb{R}$$
$$\mathbf{v}_i \in \mathbb{R}^n, \|\mathbf{v}_i\| = 1$$

$$(i = 1, \ldots, n)$$

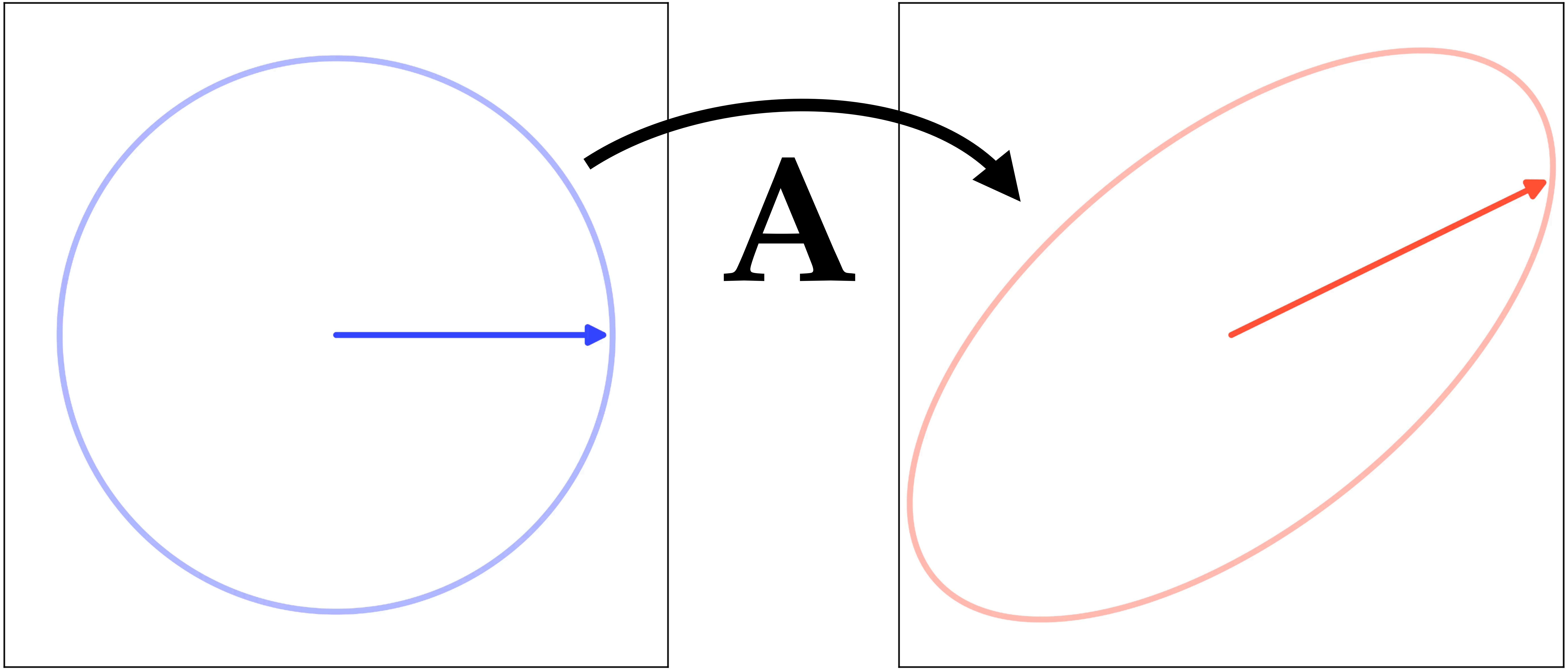
Eigendecomposition: the geometry



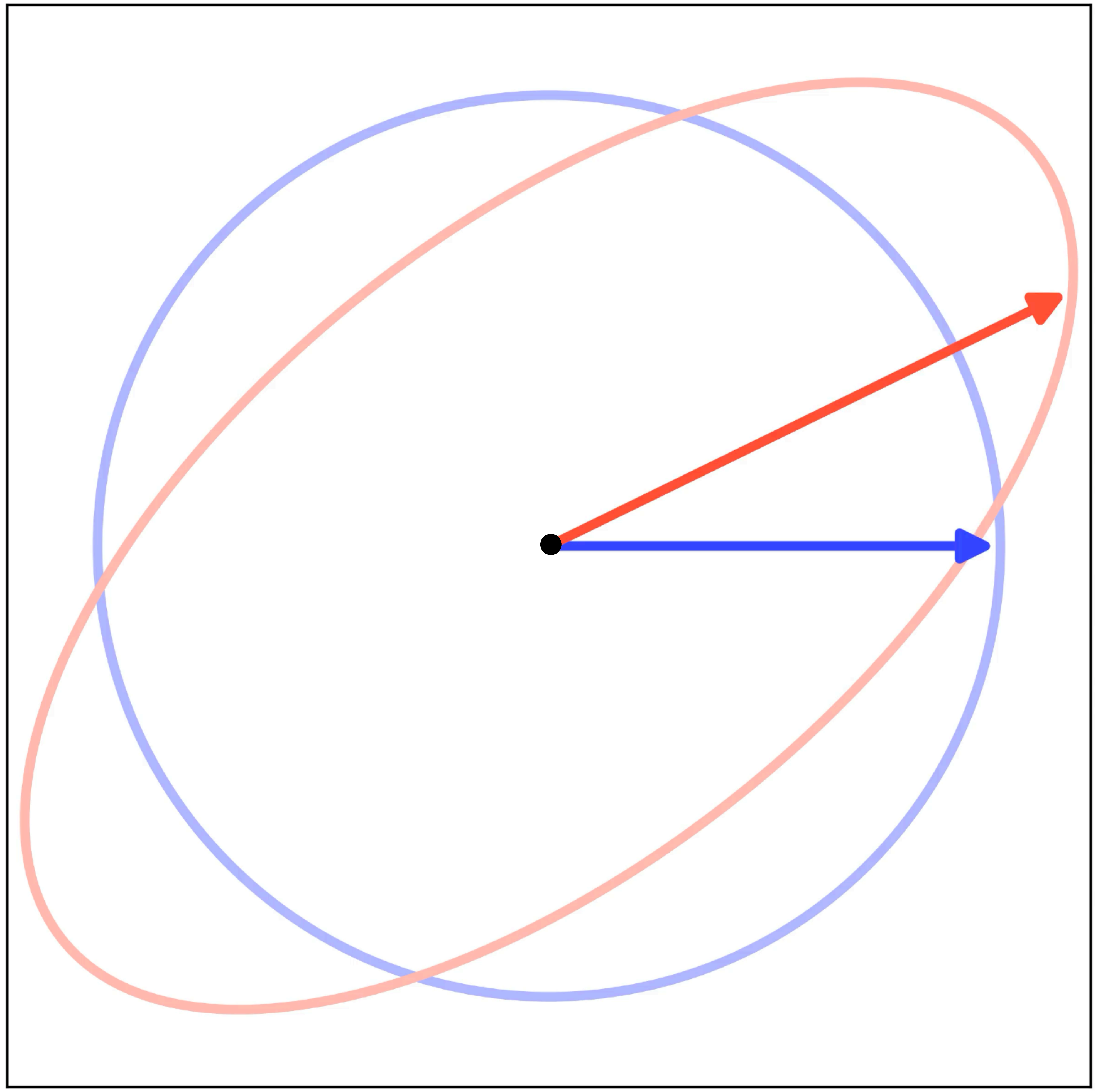
Eigendecomposition: the geometry



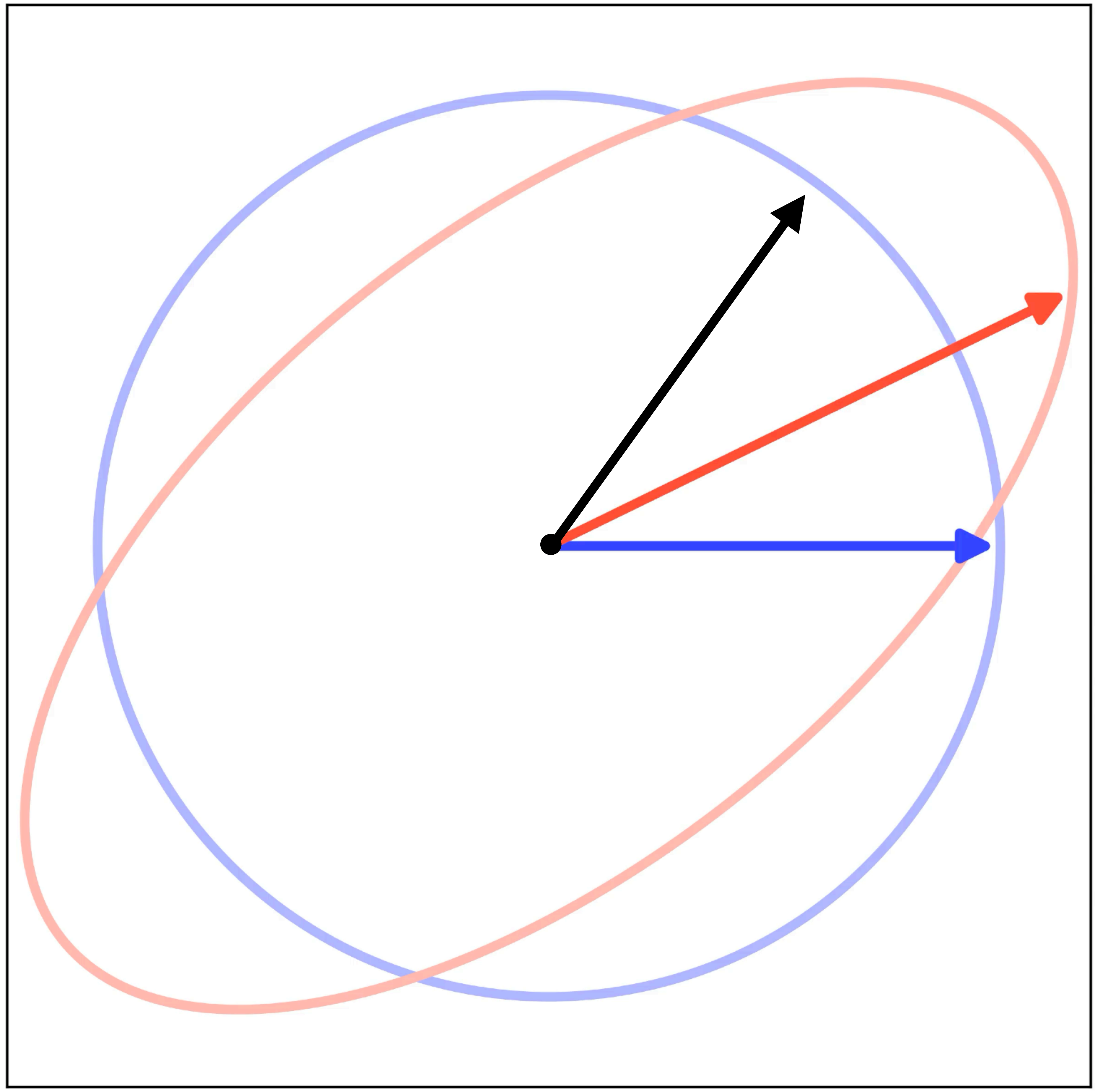
Eigendecomposition: the geometry



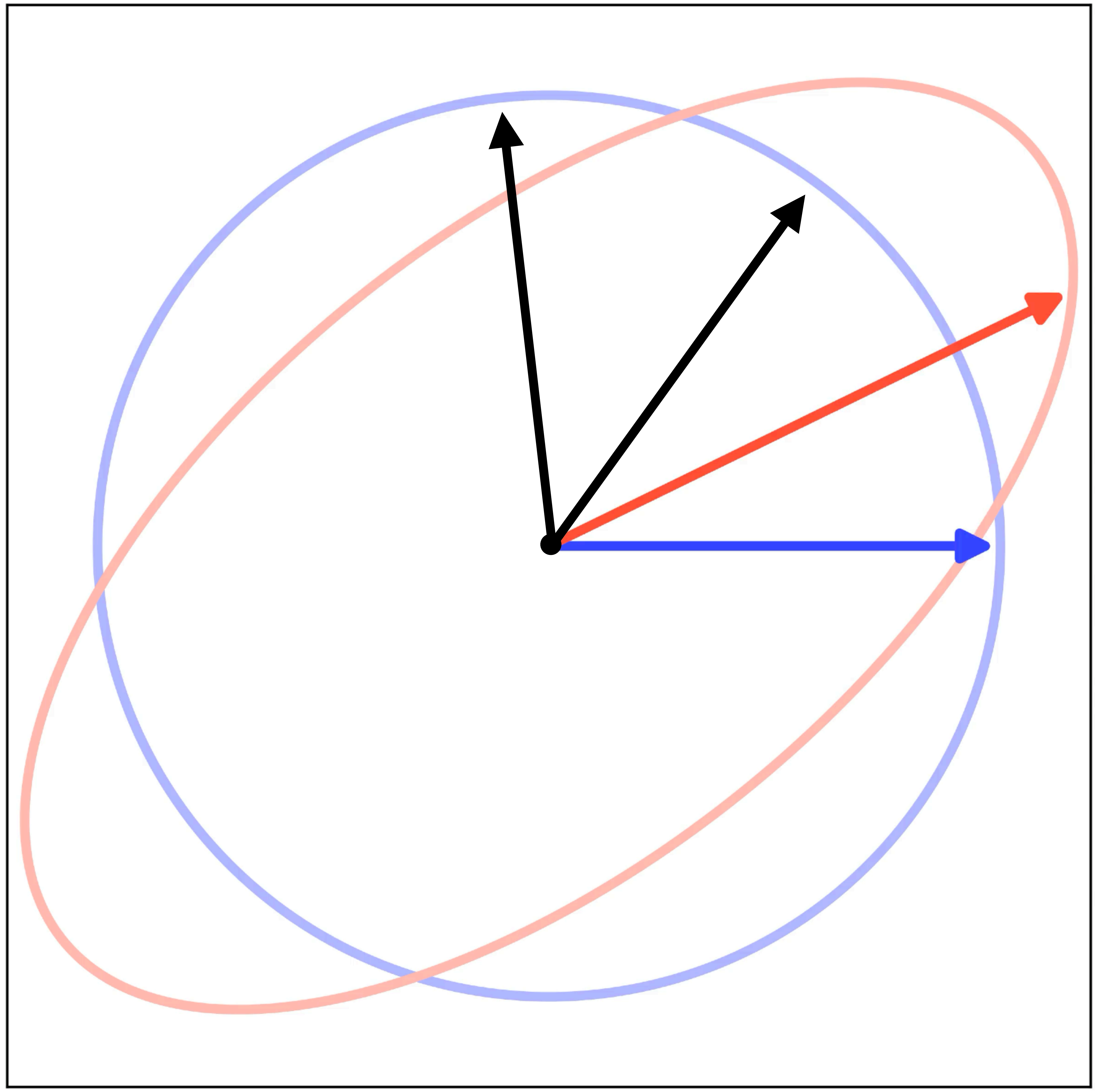
Eigendecomposition: the geometry



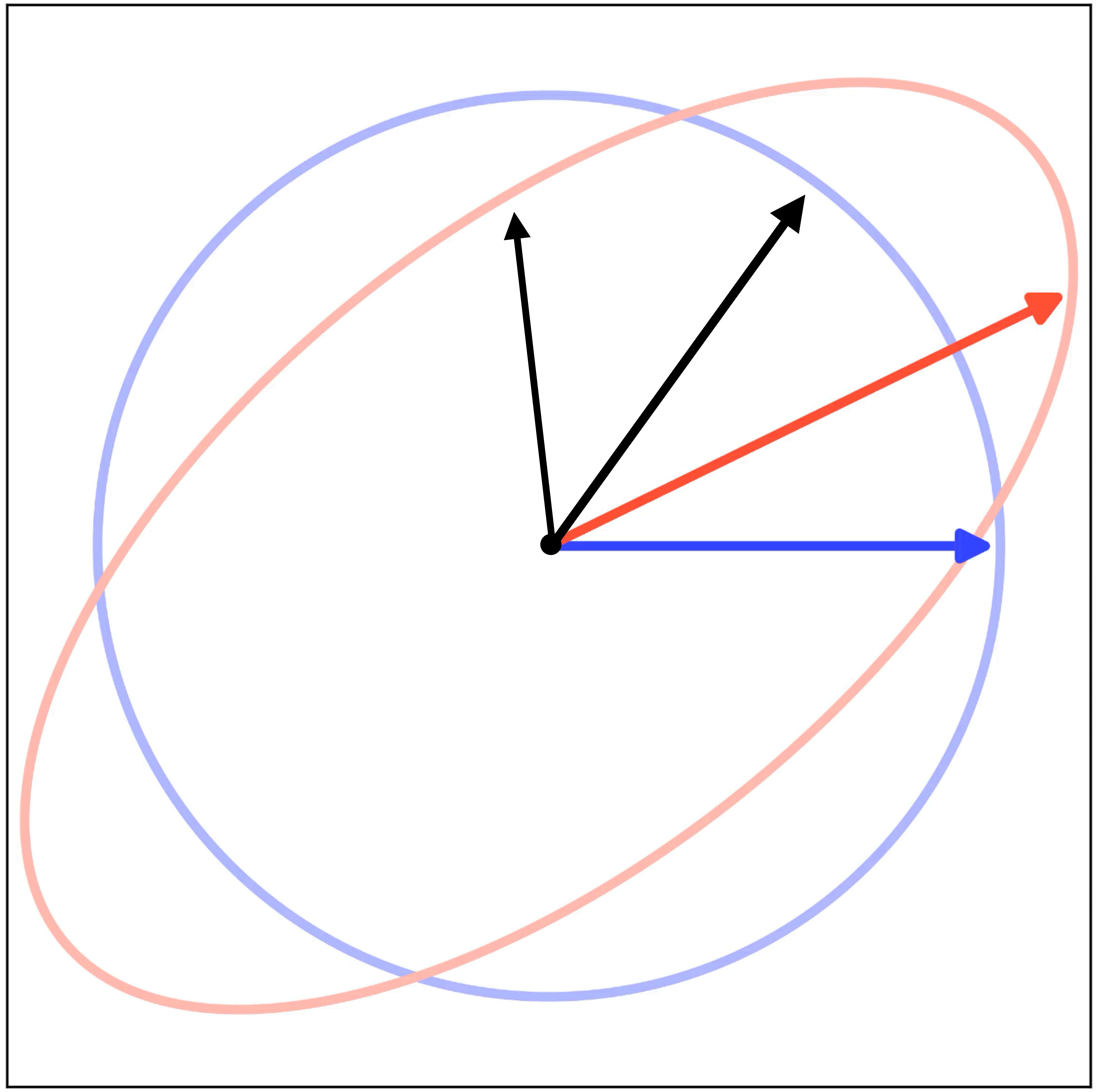
Eigendecomposition: the geometry



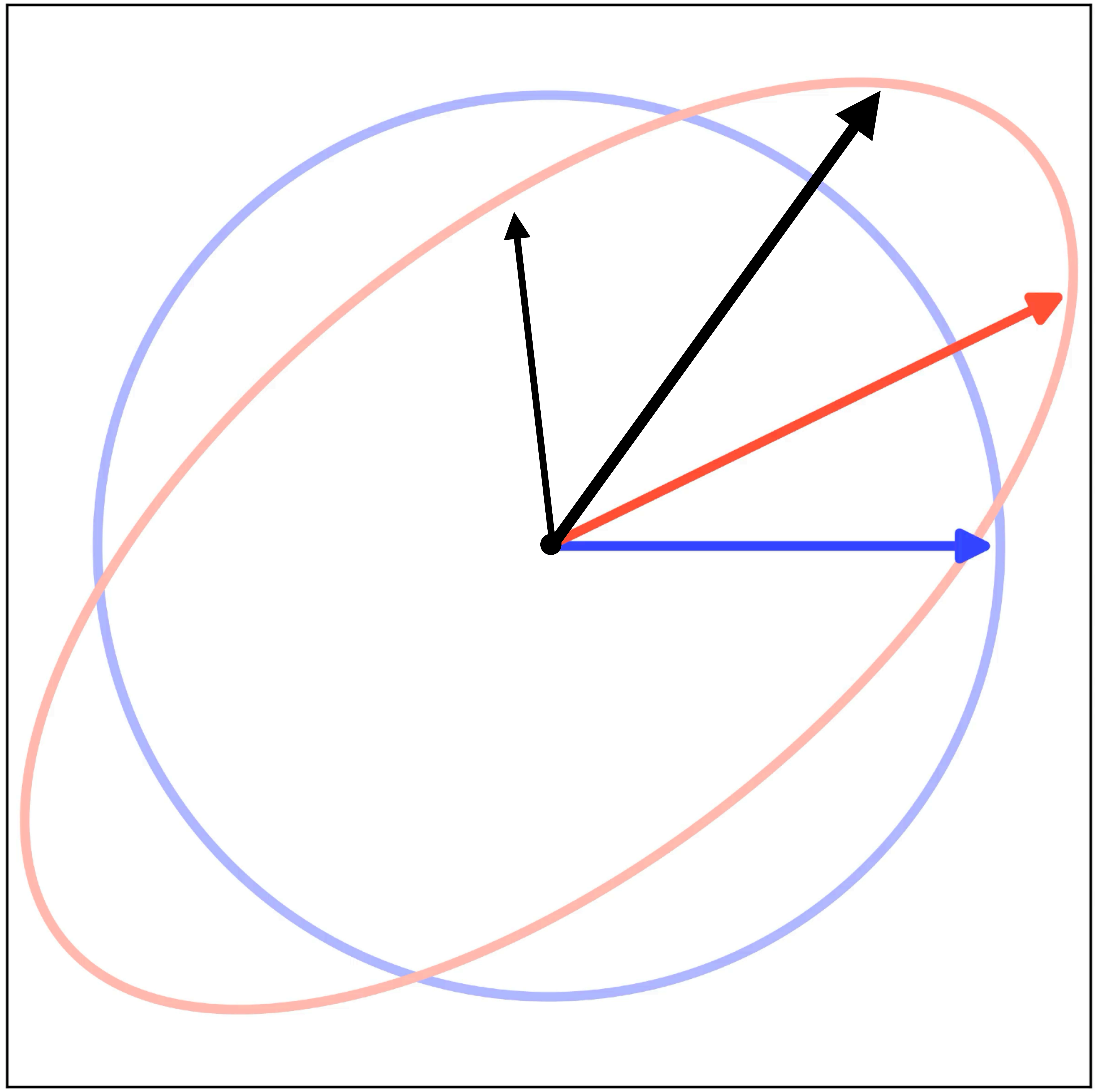
Eigendecomposition: the geometry



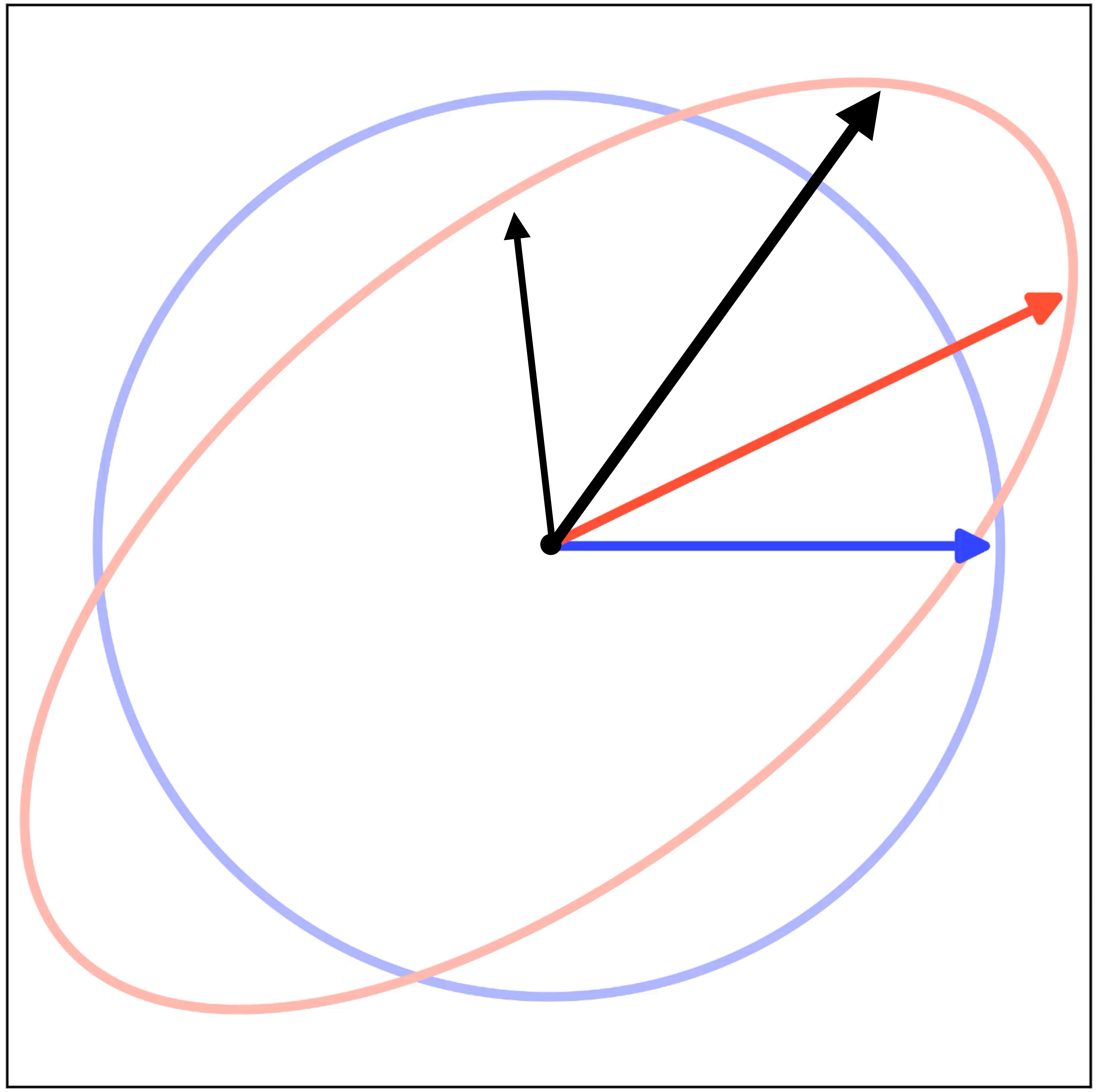
Eigendecomposition: the geometry



Eigendecomposition: the geometry

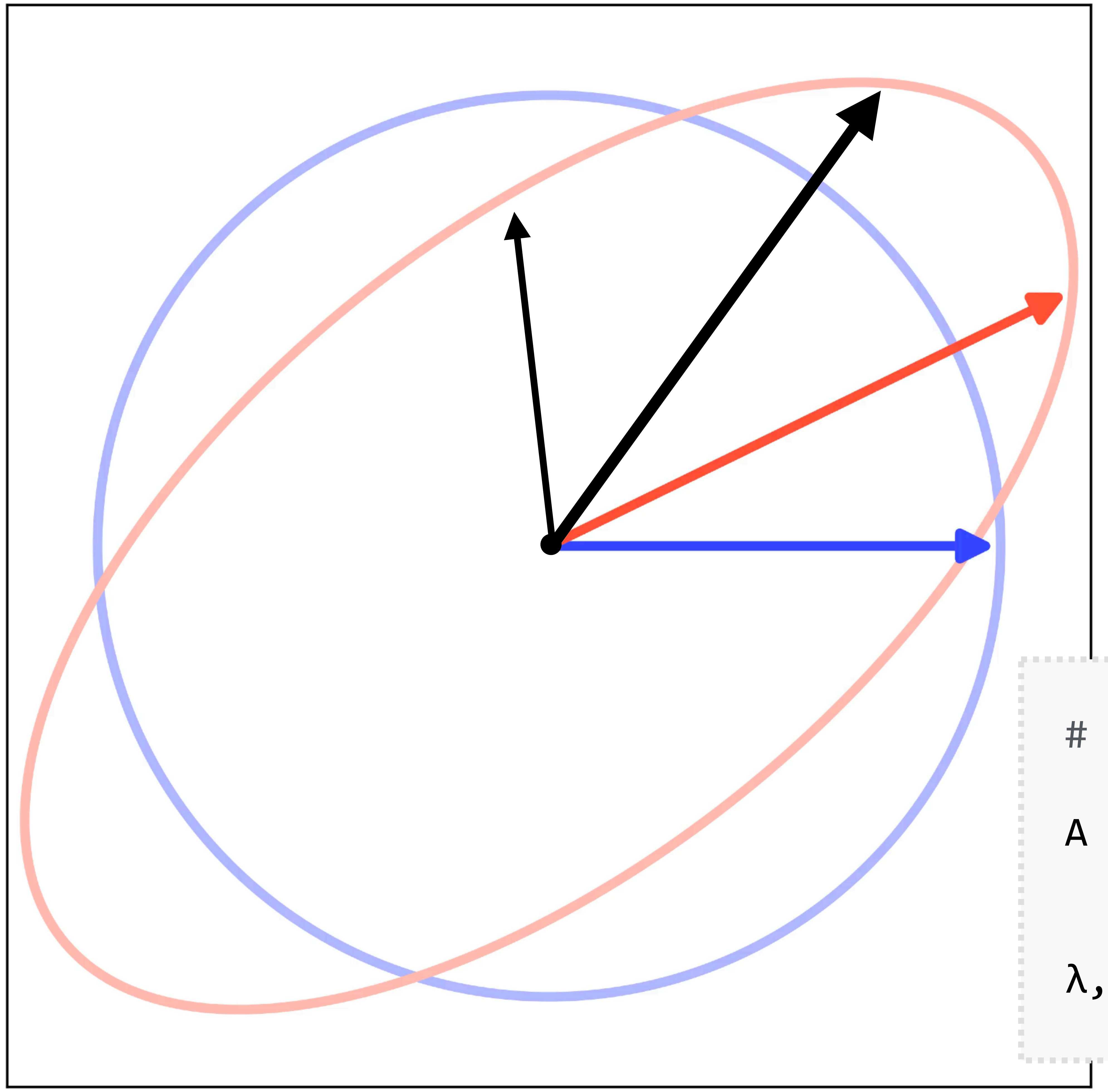


Eigendecomposition: the geometry



$$\mathbf{A}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$
$$\mathbf{A}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

Eigendecomposition: the geometry



$$\mathbf{A}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$
$$\mathbf{A}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

```
# Randomly generated matrix

A = array([[ 1.16043581,  0.0566787 ],
           [ 0.56722123,  0.85777919]])

λ, V  = scipy.linalg.eig(A)
```

Eigenvectors (.. of non-singular matrices)

- For non-singular matrices **A**:

Eigenvectors (.. of non-singular matrices)

- For non-singular matrices $\mathbf{A}$:
  - there are n eigenvalue/eigenvector pairs

Eigenvectors (.. of non-singular matrices)

- For non-singular matrices $\mathbf{A}$:
  - there are n eigenvalue/eigenvector pairs
  - the eigenvectors $\mathbf{v}_i$ form a basis for $\mathbb{R}^n$

Eigenvectors (.. of non-singular matrices)

- For non-singular matrices $\mathbf{A}$:
  - there are n eigenvalue/eigenvector pairs
  - the eigenvectors $\mathbf{v}_i$ form a basis for $\mathbb{R}^n$
    - .. which are not generally orthogonal (guaranteed only if $\mathbf{A}$ is symmetric).

Eigenvectors (.. of non-singular matrices)

- For non-singular matrices $\mathbf{A}$:
  - there are n eigenvalue/eigenvector pairs
  - the eigenvectors $\mathbf{v}_i$ form a basis for $\mathbb{R}^n$
    - .. which are not generally orthogonal (guaranteed only if $\mathbf{A}$ is symmetric).

- Therefore, a matrix with those $\mathbf{v}_i$ as its columns

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

has rank n (i.e., *full rank*) and so is non-singular

What happens when we multiply A and V?

$$\mathbf{A} \cdot \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} =$$

What happens when we multiply A and V?

$$\mathbf{A} \cdot \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \lambda_1 \mathbf{v}_1 & \cdots & \lambda_n \mathbf{v}_n \\ | & & | \end{pmatrix}$$

What happens when we multiply A and V?

$$\mathbf{A} \cdot \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \lambda_1 \mathbf{v}_1 & \cdots & \lambda_n \mathbf{v}_n \\ | & & | \end{pmatrix} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

What happens when we multiply A and V?

$$\mathbf{A} \cdot \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \lambda_1 \mathbf{v}_1 & \cdots & \lambda_n \mathbf{v}_n \\ | & & | \end{pmatrix} = \mathbf{V} \underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}}_{=:\mathbf{\Lambda}}$$

What happens when we multiply A and V?

$$\mathbf{A} \cdot \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \lambda_1 \mathbf{v}_1 & \cdots & \lambda_n \mathbf{v}_n \\ | & & | \end{pmatrix} = \mathbf{V} \underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}}_{=:\mathbf{\Lambda}}$$

In other words:

$$\mathbf{AV} = \mathbf{V\Lambda}$$

What happens when we multiply A and V?

$$\mathbf{A}\cdot\begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \lambda_1\mathbf{v}_1 & \cdots & \lambda_n\mathbf{v}_n \\ | & & | \end{pmatrix} = \mathbf{V}\underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}}_{=:\mathbf{\Lambda}}$$

In other words:

The Eigendecomposition

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

Geometric summary

The eigendecomp. expresses a linear transformation $\mathbf{A}$ as product of 3 transforms

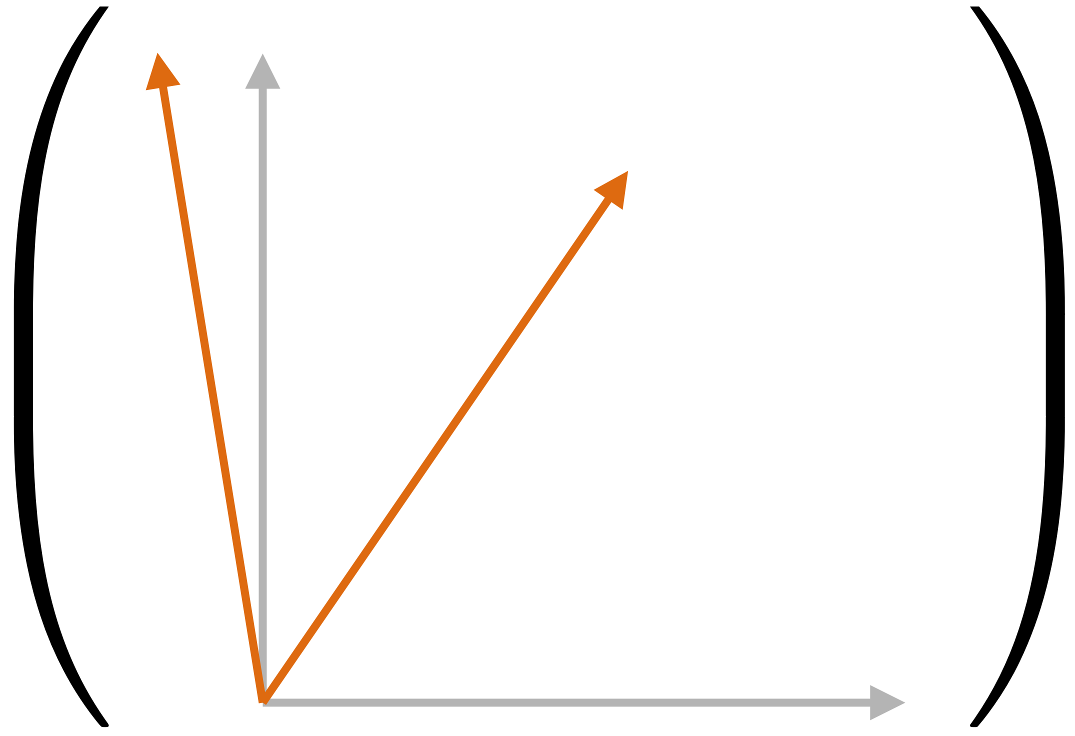
$$\mathbf{V}^{-1}$$

$$\mathbf{A} =$$

Geometric summary

The eigendecomp. expresses a linear transformation $\mathbf{A}$ as product of 3 transforms

$$\mathbf{A} = \qquad \mathbf{V}^{-1}$$

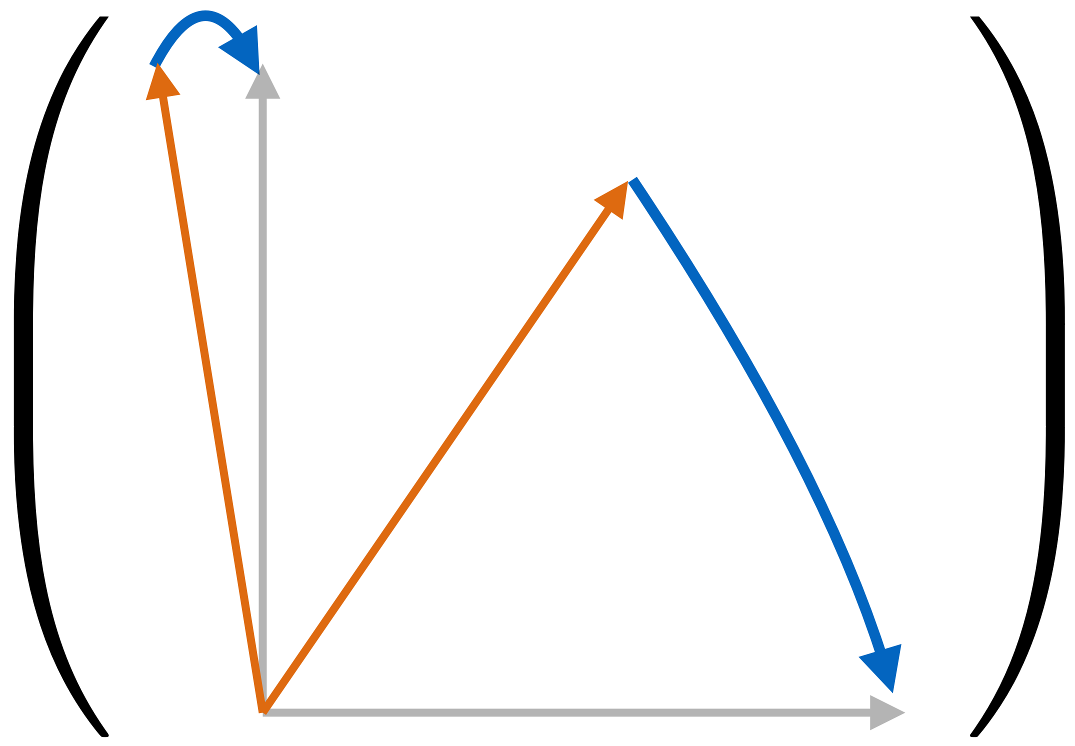


change of basis (*not necessarily orthogonal!*)

Geometric summary

The eigendecomp. expresses a linear transformation $\mathbf{A}$ as product of 3 transforms

$$\mathbf{A} = \qquad \mathbf{V}^{-1}$$

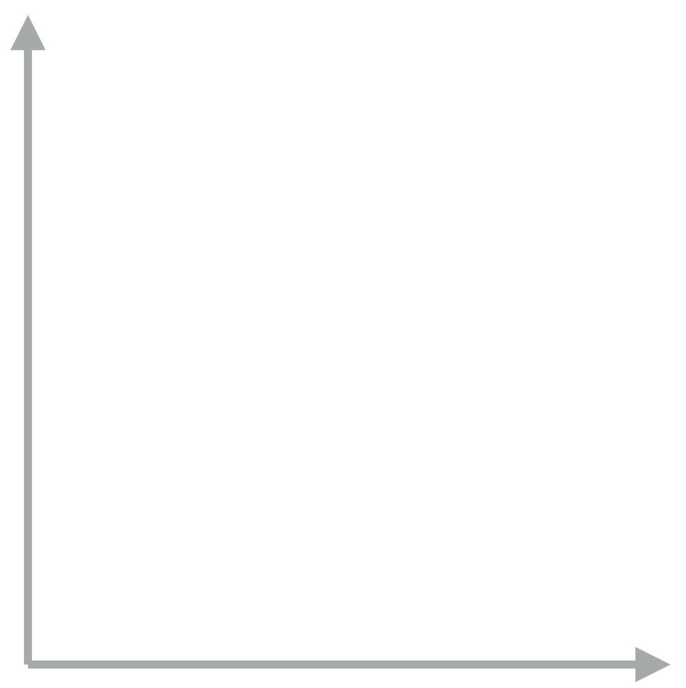


change of basis (*not necessarily orthogonal!*)

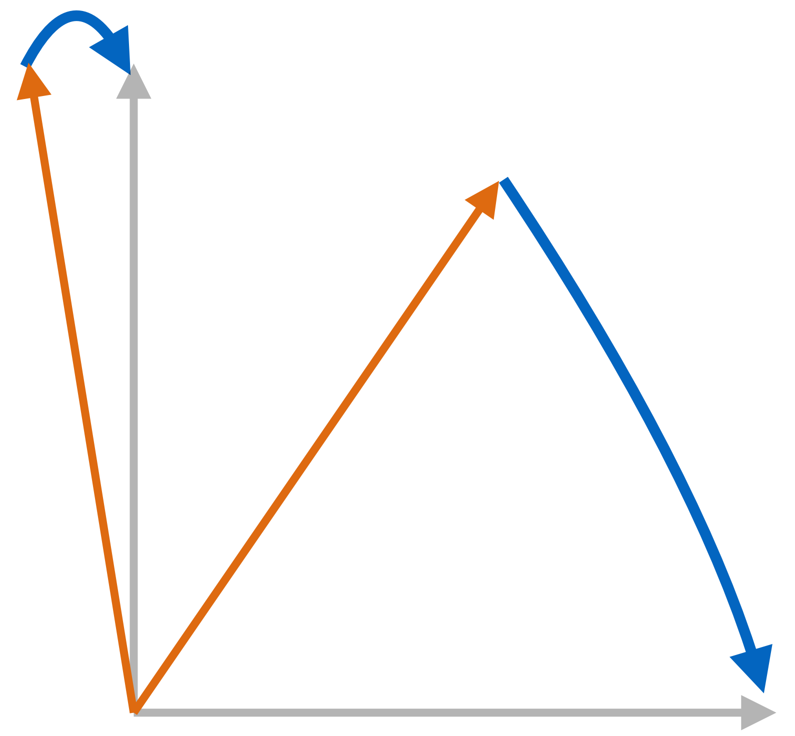
Geometric summary

The eigendecomp. expresses a linear transformation $\mathbf{A}$ as product of 3 transforms

$$\mathbf{A} = \boldsymbol{\Lambda} \cdot \mathbf{V}^{-1}$$



non-uniform scale

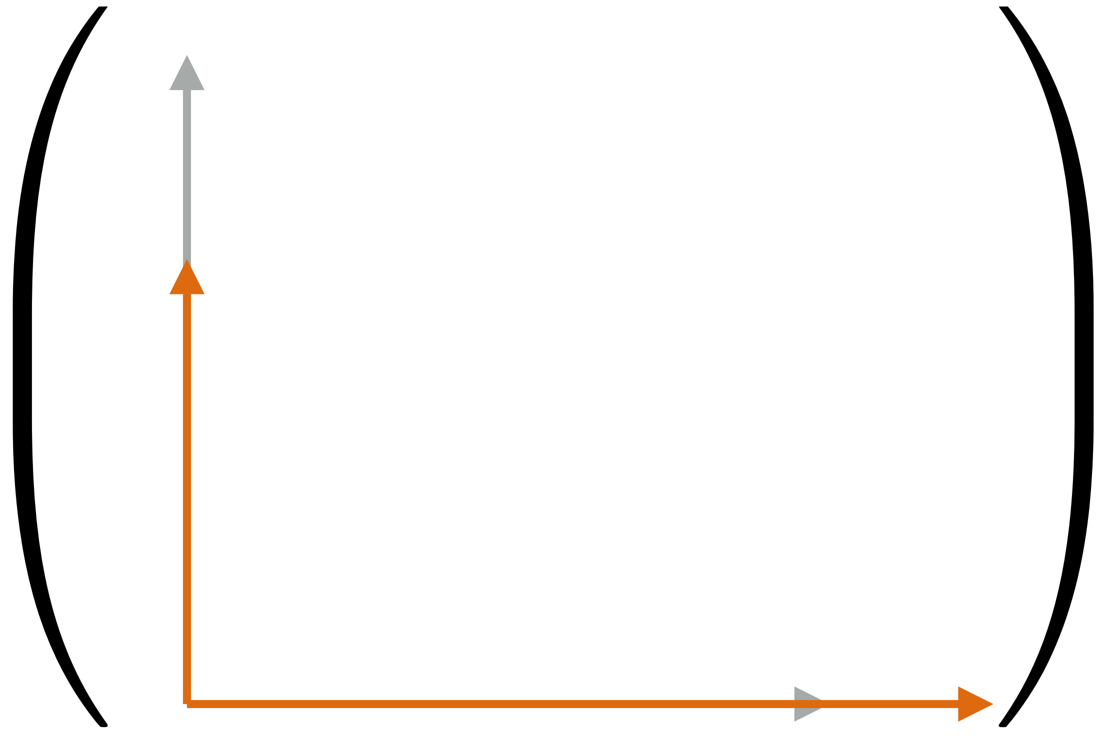


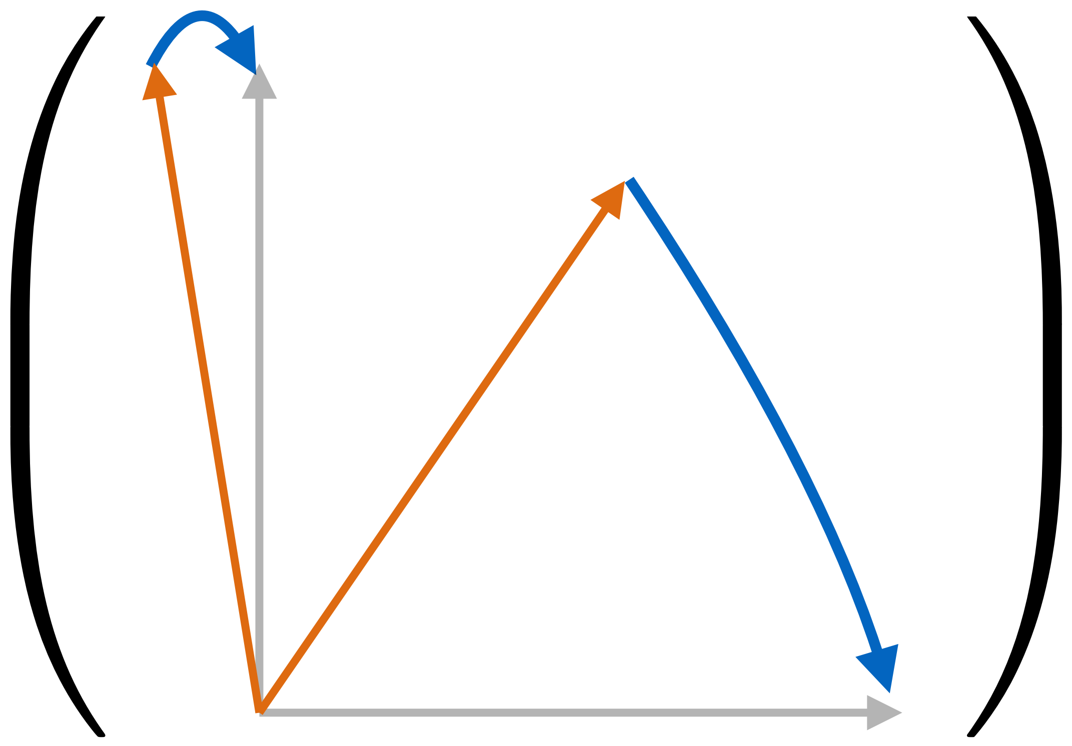
change of basis (*not necessarily orthogonal!*)

Geometric summary

The eigendecomp. expresses a linear transformation A as product of 3 transforms

$$\mathbf{A} = \quad \mathbf{\Lambda} \cdot \mathbf{V}^{-1}$$





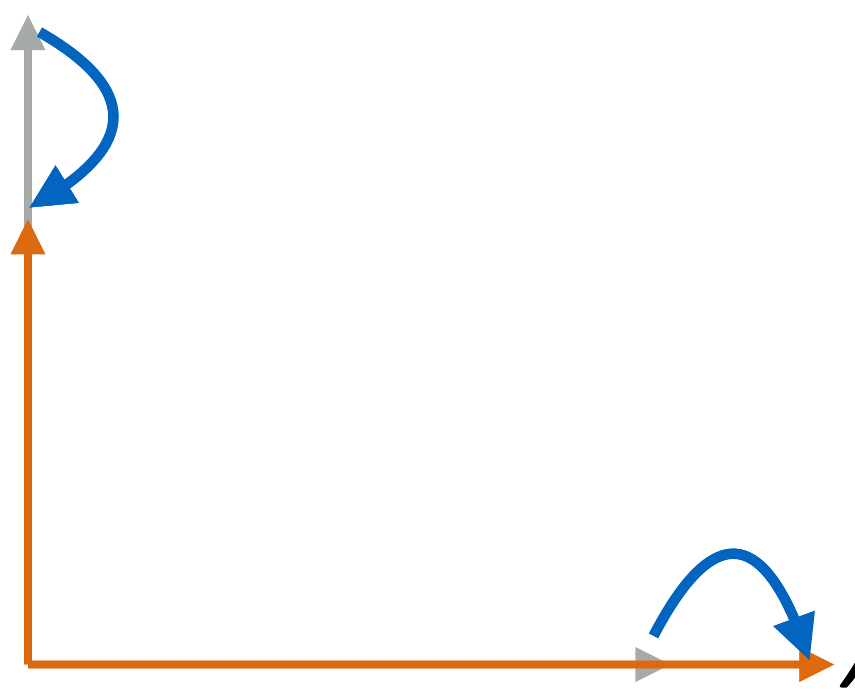
non-uniform scale

change of basis (*not necessarily orthogonal!*)

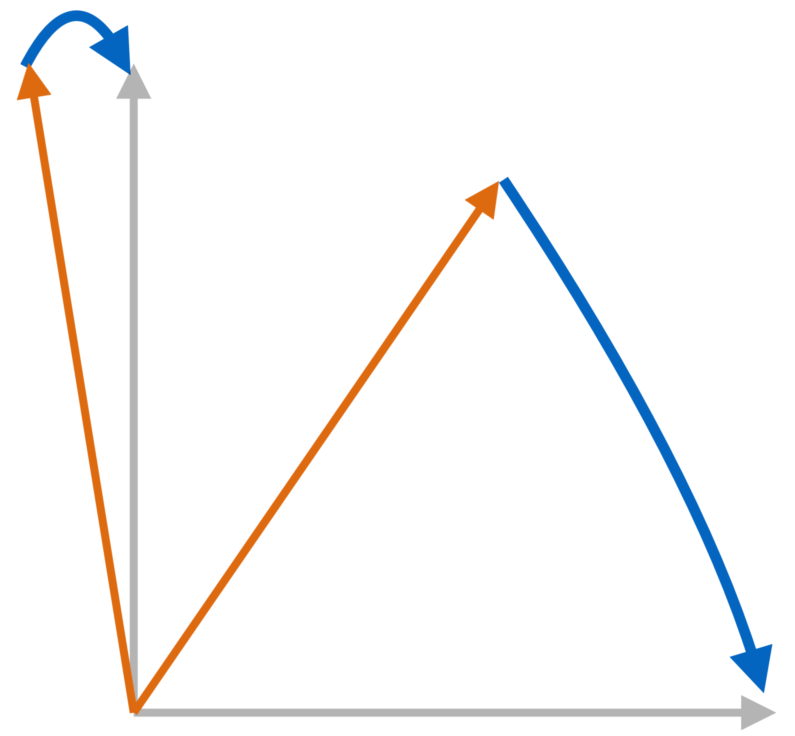
Geometric summary

The eigendecomp. expresses a linear transformation A as product of 3 transforms

$$\mathbf{A} = \qquad \mathbf{\Lambda} \qquad \mathbf{V}^{-1}$$



$\cdot$



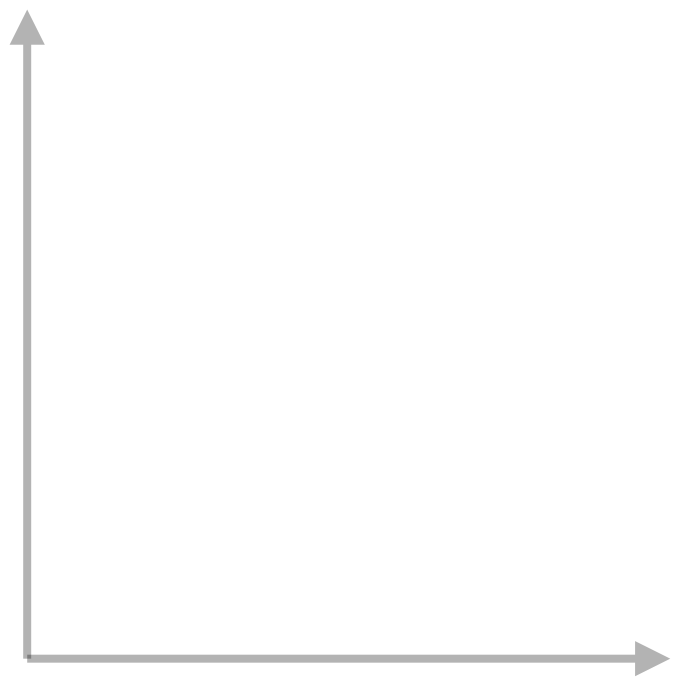
non-uniform scale

change of basis (*not necessarily orthogonal!*)

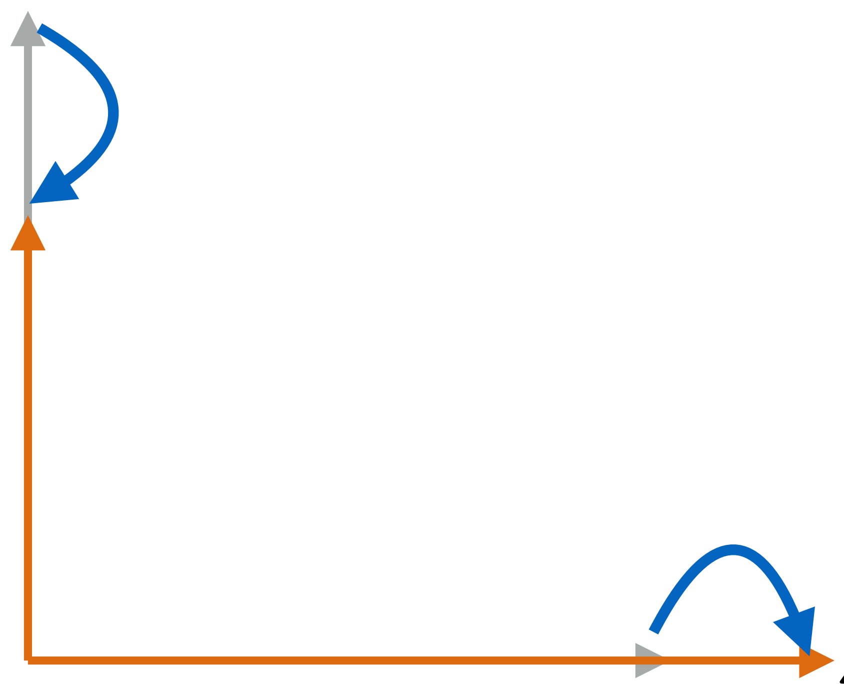
Geometric summary

The eigendecomp. expresses a linear transformation A as product of 3 transforms

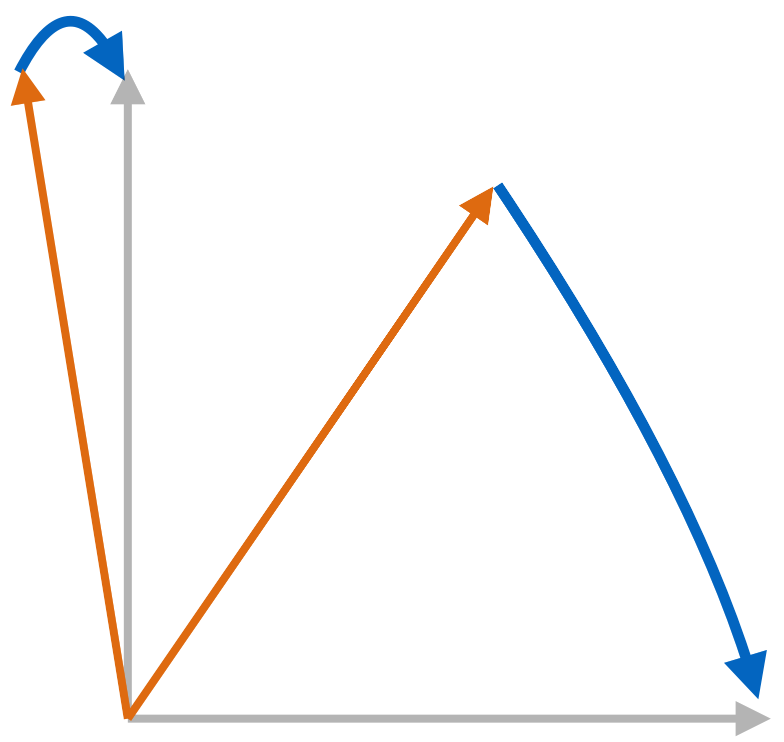
$$\mathbf{A} = \mathbf{V} \cdot \boldsymbol{\Lambda} \cdot \mathbf{V}^{-1}$$



(inverse)
change of basis



non-uniform scale

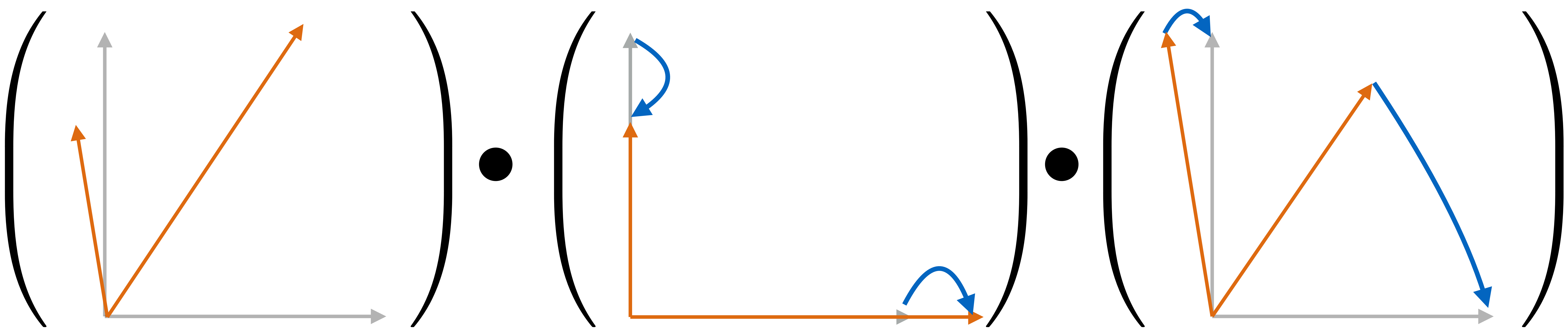


change of basis (*not necessarily orthogonal!*)

Geometric summary

The eigendecomp. expresses a linear transformation A as product of 3 transforms

$$\mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1}$$



(inverse) change of basis

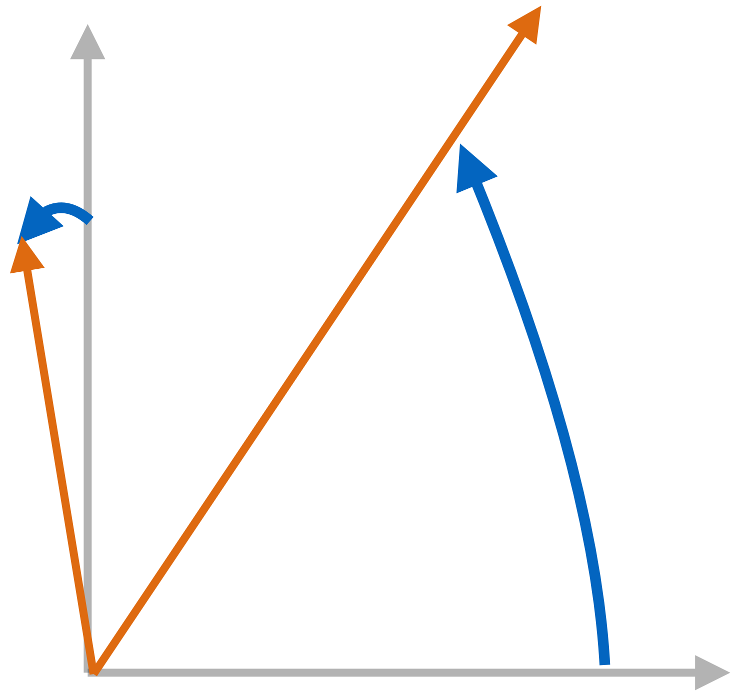
non-uniform scale

change of basis (*not necessarily orthogonal!*)

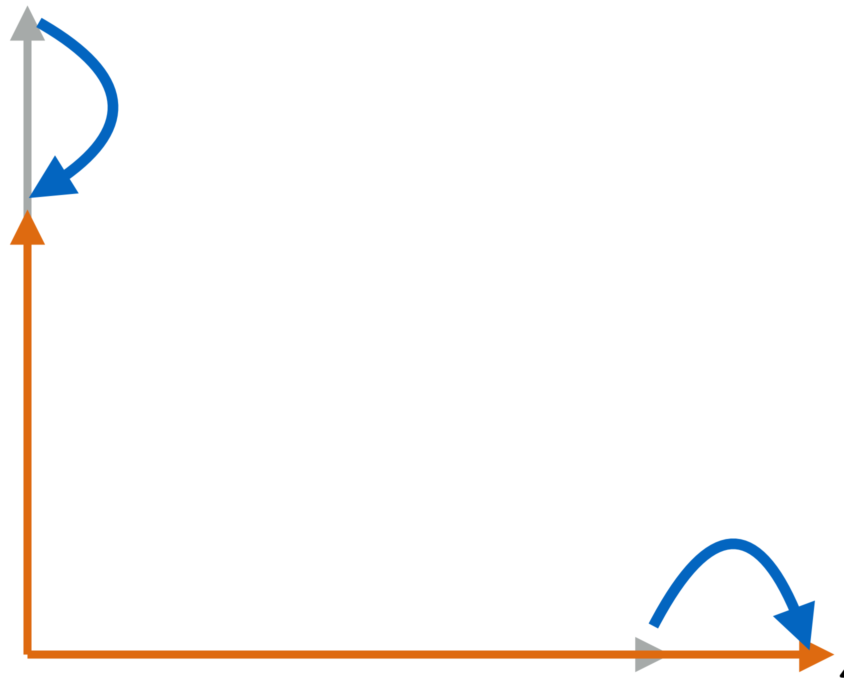
Geometric summary

The eigendecomp. expresses a linear transformation $\mathbf{A}$ as product of 3 transforms

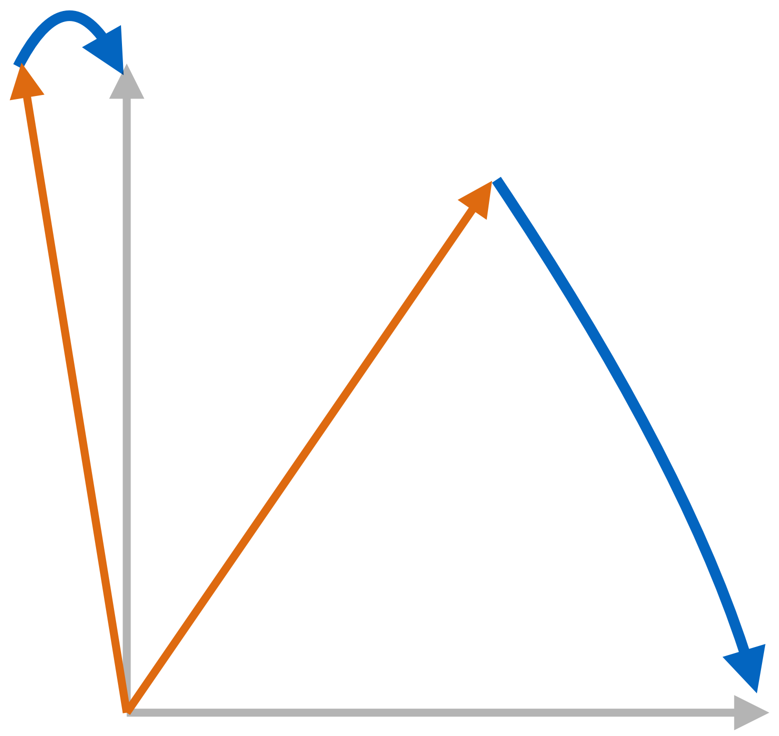
$$\mathbf{A} = \mathbf{V} \cdot \boldsymbol{\Lambda} \cdot \mathbf{V}^{-1}$$



(inverse) change of basis



non-uniform scale



change of basis (*not necessarily orthogonal!*)

Eigendecomposition: what is it good for?

Eigendecomposition: what is it good for?

- Taking integer & fractional powers of matrices.

Eigendecomposition: what is it good for?

- **Taking integer & fractional powers of matrices.**
- **Solving linear ordinary differential equations (ODEs).**
  - Physics models the world using differential equations.

Eigendecomposition: what is it good for?

- **Taking integer & fractional powers of matrices.**
- **Solving linear ordinary differential equations (ODEs).**
  - Physics models the world using differential equations.
- **Estimating covariance matrices.**
  - Normal distribution: *the* central statistical distribution.
    Eigendecomposition "fits" normal distribution to data points.

Eigendecomposition: what is it good for?

- **Taking integer & fractional powers of matrices.**
- **Solving linear ordinary differential equations (ODEs).**
  - Physics models the world using differential equations.
- **Estimating covariance matrices.**
  - Normal distribution: *the* central statistical distribution. Eigendecomposition "fits" normal distribution to data points.
- **Dimensionality reduction.**
  - Important processing step to remove irrelevant data.

Eigendecomposition: what is it good for?

- **Taking integer & fractional powers of matrices.**
- **Solving linear ordinary differential equations (ODEs).**
  - Physics models the world using differential equations.
- **Estimating covariance matrices.**
  - Normal distribution: *the* central statistical distribution. Eigendecomposition "fits" normal distribution to data points.
- **Dimensionality reduction.**
  - Important processing step to remove irrelevant data.

Will see examples of all of these.
(Some in the context of SVD)

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det\left(\lambda\mathbf{I} - \mathbf{A}\right) = 0.$$

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det\left(\lambda \mathbf{I} - \mathbf{A}\right) = 0.$$

- As the name indicates, this is a polynomial.

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det\left(\lambda\mathbf{I} - \mathbf{A}\right) = 0.$$

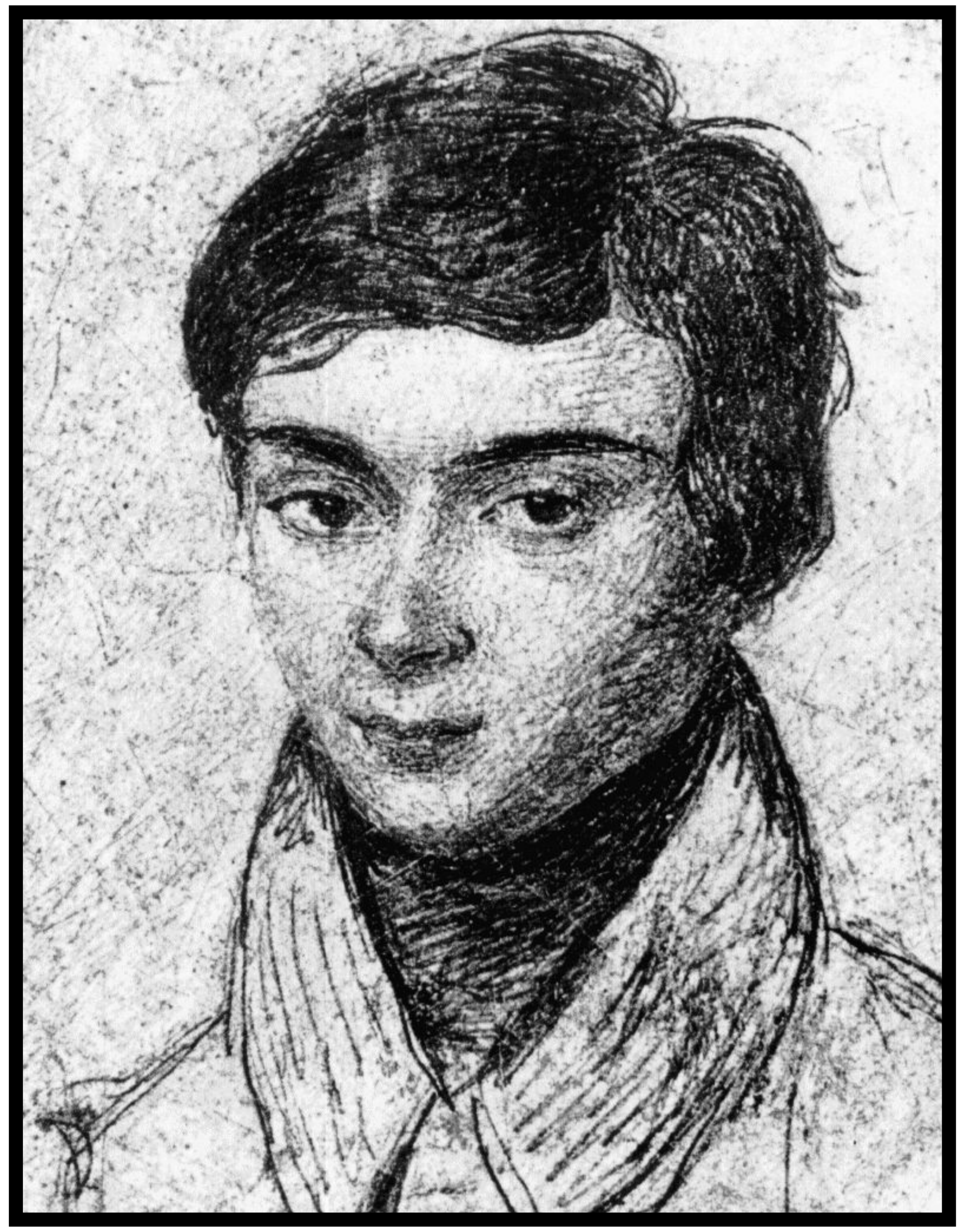
- As the name indicates, this is a polynomial.
- Degree n for a $n \times n$ matrix

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det\left(\lambda\mathbf{I} - \mathbf{A}\right) = 0.$$

- As the name indicates, this is a polynomial.
- Degree n for a $n \times n$ matrix
- *Galois theory*: most polynomials of degree 5 or higher cannot be solved using *algebraic methods*.



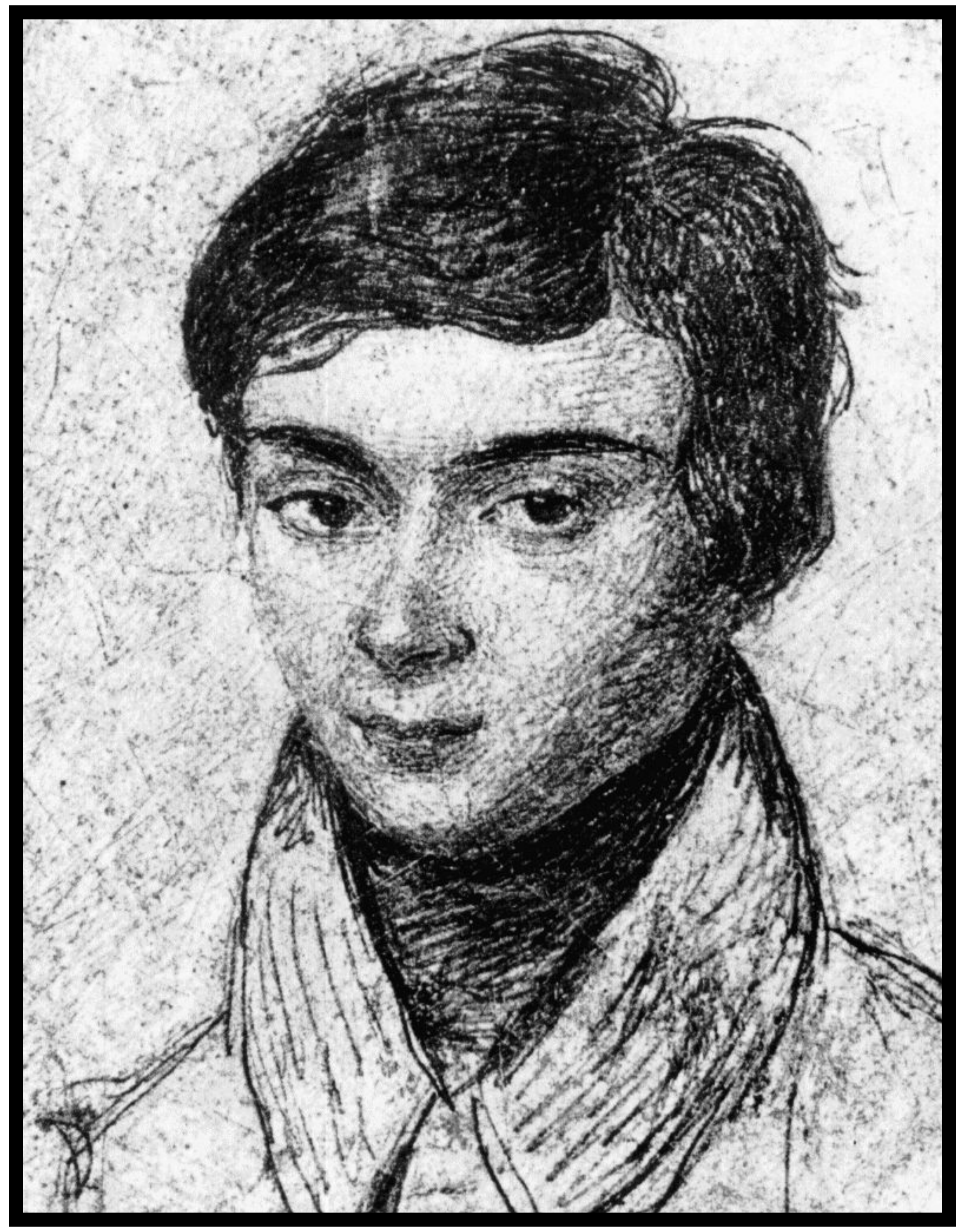
Évariste Galois (1811-1832)

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0.$$

- As the name indicates, this is a polynomial.
- Degree n for a $n \times n$ matrix
- *Galois theory*: most polynomials of degree 5 or higher cannot be solved using *algebraic methods*.
- Oops.



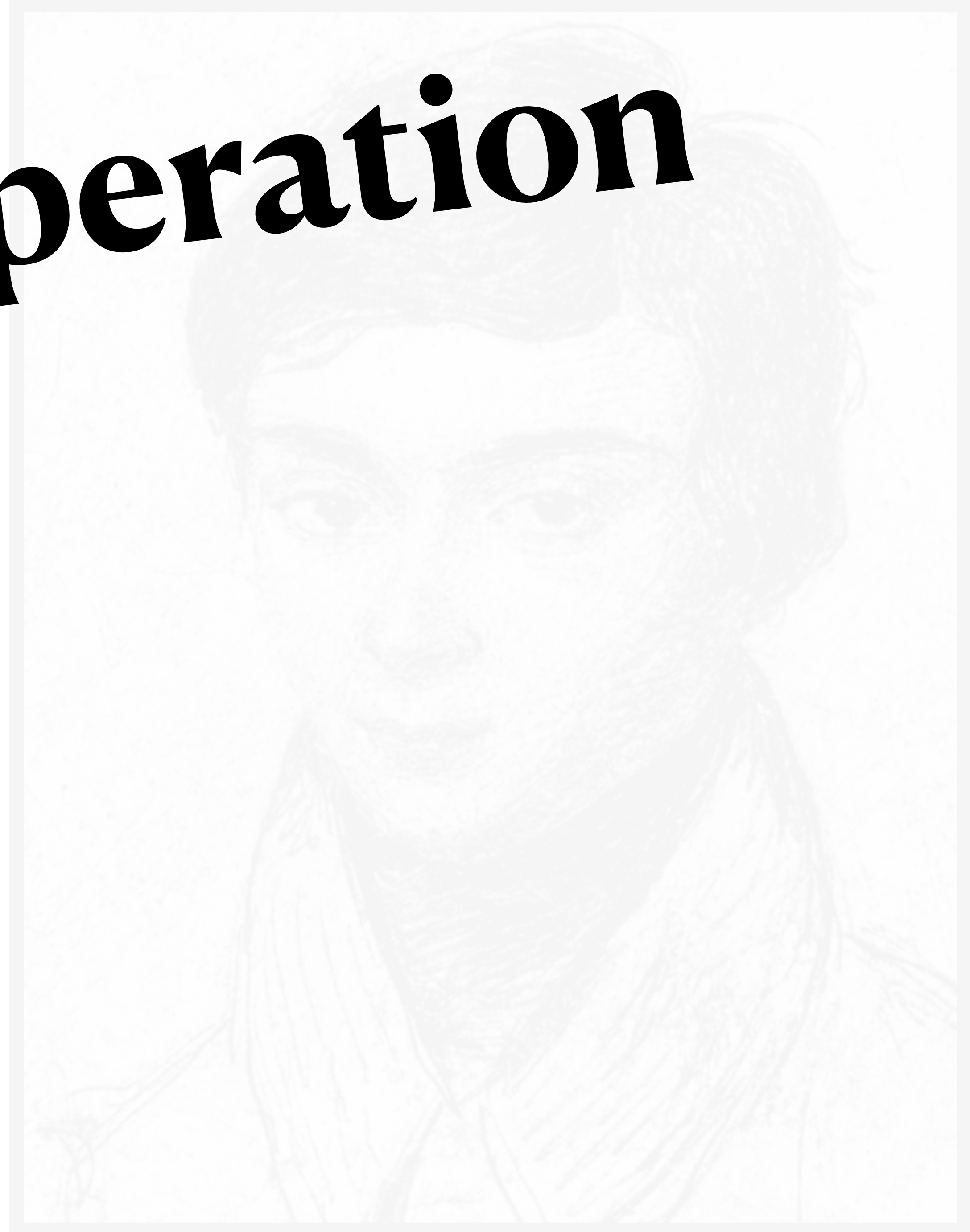
Évariste Galois (1811-1832)

How to find the eigendecomposition?

MATH-111: Use the Characteristic Polynomial!

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0$$

- As the name indicates, this is a polynomial.
- Degree n for $n \times n$ matrix
- *Galois theory*: most polynomials of degree 5 or higher cannot be solved using *algebraic methods*.
- Oops



Évariste Galois (1811-1832)

Eigendecomposition is a **nonlinear operation** (our very first one!)

- Approximate result
- No guarantees on computation time

Matrix powers

The following works for both integer and fractional powers

- Let's compute

Matrix powers

The following works for both integer and fractional powers

- Let's compute

$$\mathbf{A}^2 =$$

Matrix powers

The following works for both integer and fractional powers

- Let's compute

$$\mathbf{A}^2 = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

Matrix powers

The following works for both integer and fractional powers

- Let's compute

$$\mathbf{A}^2 = \mathbf{V}\boldsymbol{\Lambda}\boldsymbol{\Lambda}\mathbf{V}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^2\mathbf{V}^{-1}$$

Matrix powers

The following works for both integer and fractional powers

- Let's compute

$$\mathbf{A}^2 = \mathbf{V}\mathbf{\Lambda}\mathbf{\Lambda}\mathbf{V}^{-1} = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^{-1}$$

- General rule:

$$\mathbf{A}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

Matrix powers

The following works for both integer and fractional powers

- Let's compute

$$\mathbf{A}^2 = \mathbf{V}\boldsymbol{\Lambda}\boldsymbol{\Lambda}\mathbf{V}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^2\mathbf{V}^{-1}$$

- General rule:

$$\mathbf{A}^k = \mathbf{V}\boldsymbol{\Lambda}^k\mathbf{V}^{-1}$$

- That includes the inverse as well:

$$\mathbf{A}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^{-1}$$

Observation when taking high powers of a matrix

(Building on the power identity from the previous slide)

$$\mathbf{A}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

Observation when taking high powers of a matrix

(Building on the power identity from the previous slide)

$$\mathbf{A}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

$$\mathbf{\Lambda}^k = \begin{pmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{pmatrix}$$

As $k \to \infty$, first entry becomes **huge** compared to others.

Observation when taking high powers of a matrix

(Building on the power identity from the previous slide)

$$\mathbf{A}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

$$\mathbf{\Lambda}^k \approx \begin{pmatrix} \lambda_1^k & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix}$$

As $k \to \infty$, first entry becomes **huge** compared to others.

Power iteration

A simple method to compute the dominant eigenvector

Power iteration

A simple method to compute the dominant eigenvector

Power iteration:

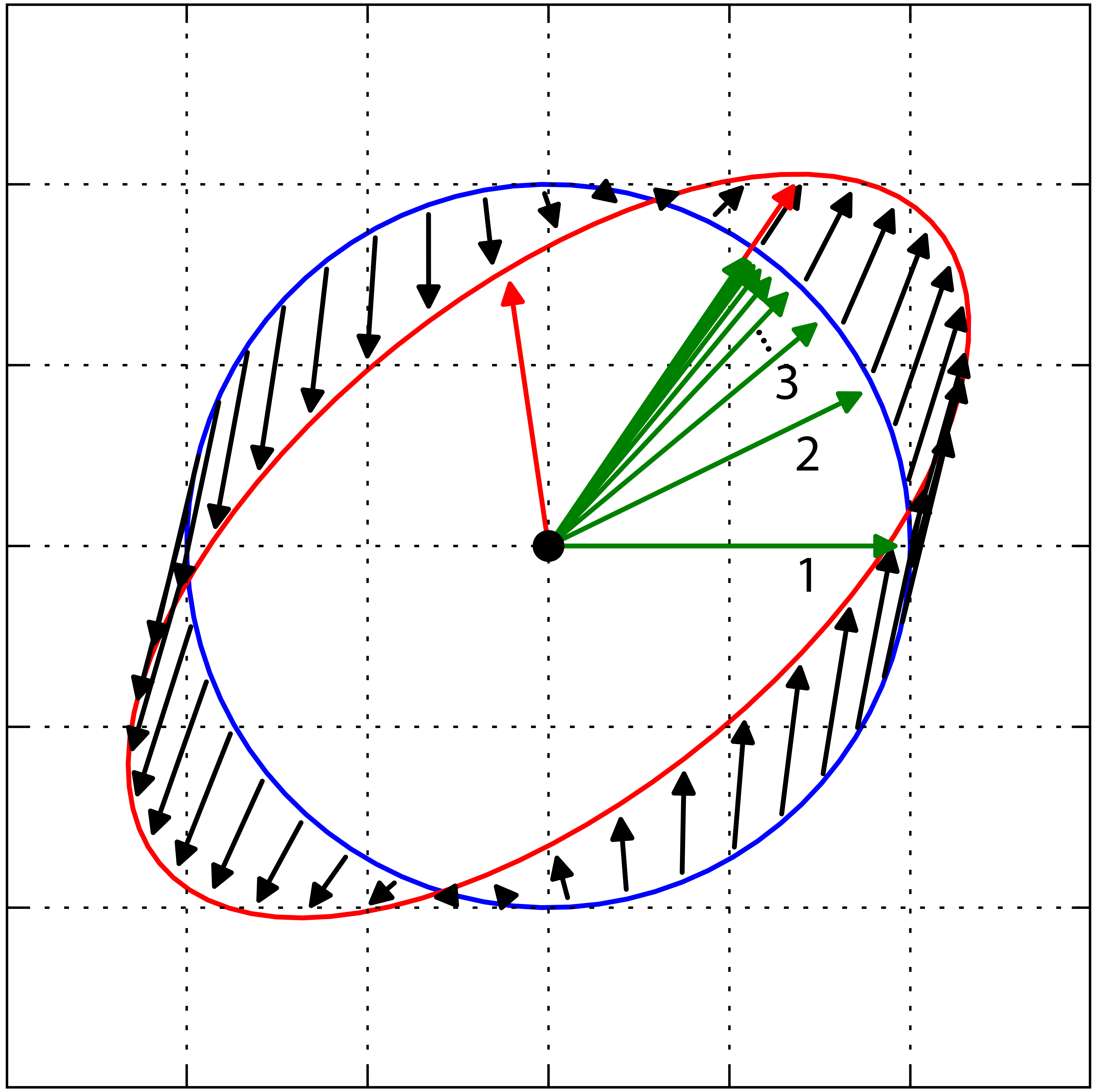
```
x = np.random.random(size)

for i in range(num_iterations):
    x = A @ x
    x /= np.linalg.norm(x)
```

Power iteration

A simple method to compute the dominant eigenvector

Power iteration:

```
x = np.random.random(size)

for i in range(num_iterations):
    x = A @ x
    x /= np.linalg.norm(x)
```



Power iteration

Power iteration

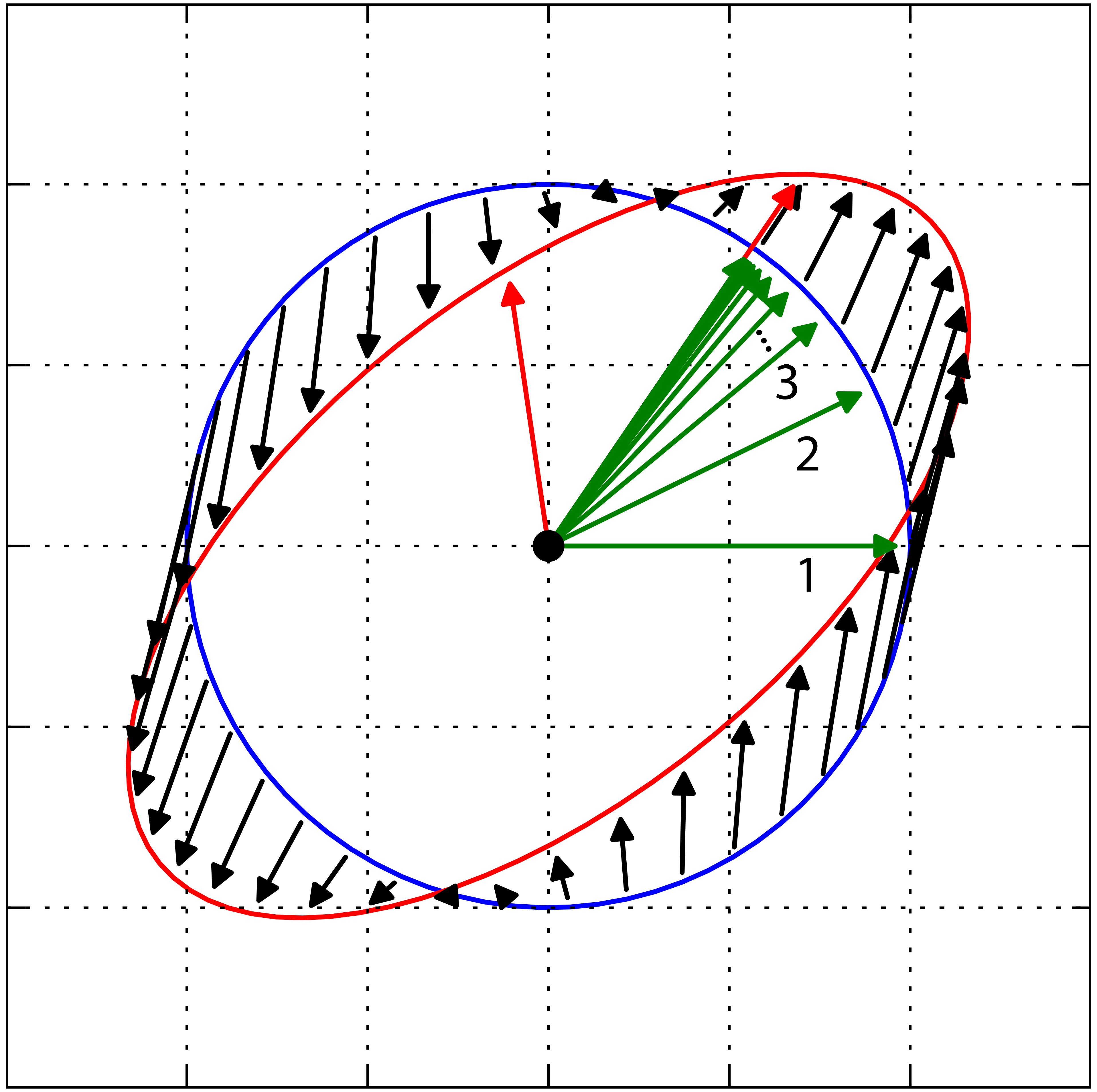
A simple method to compute the dominant eigenvector

Power iteration:

```
x = np.random.random(size)

for i in range(num_iterations):
    x = A @ x
    x /= np.linalg.norm(x)
```

Inverse iteration:

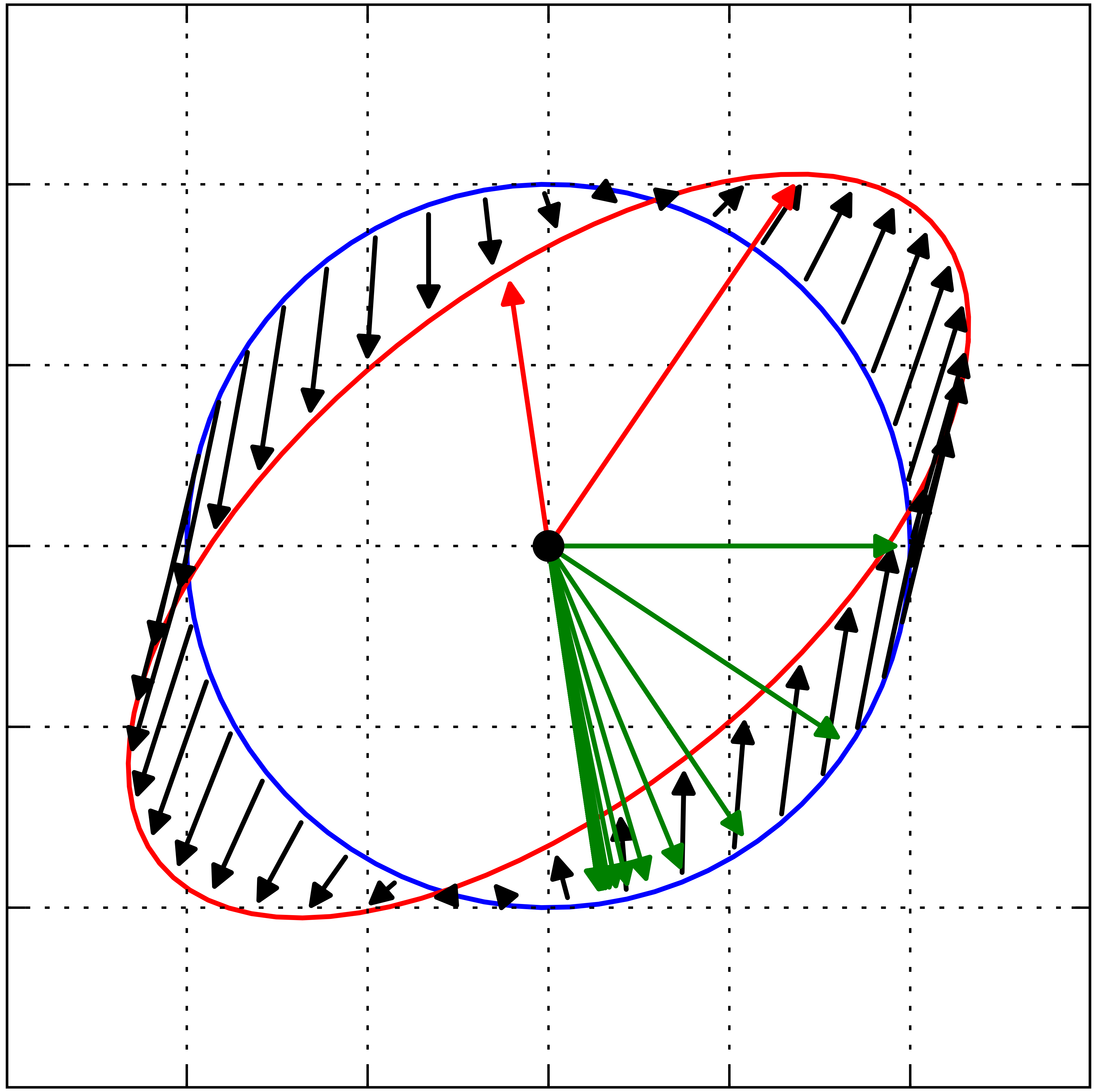
```
x = np.random.random(size)
A_lup = la.lu_factor(A)

for i in range(num_iterations):
    x = la.lu_solve(A_lup, x)
    x /= np.linalg.norm(x)
```



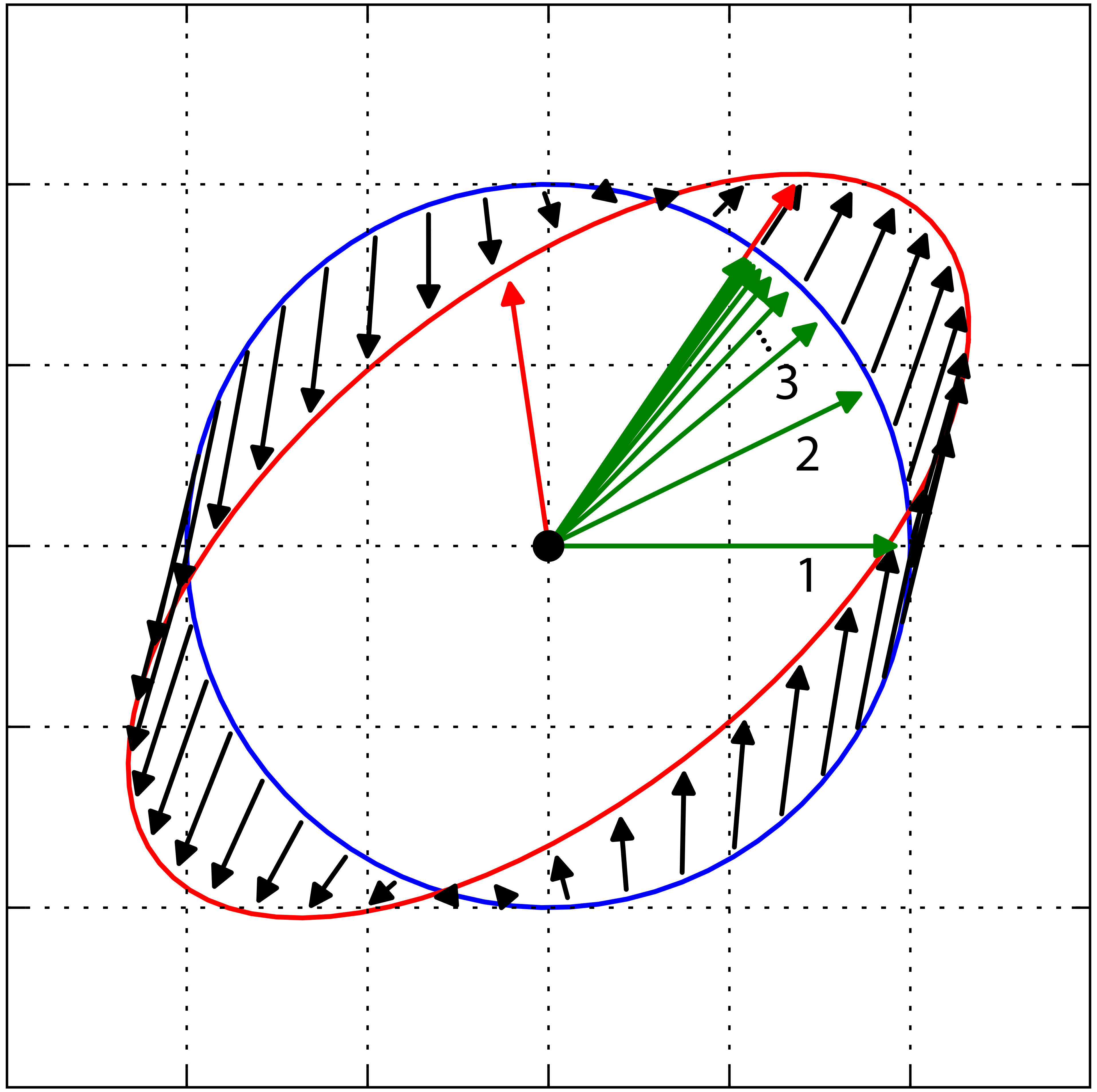
Power iteration

Power iteration

A simple method to compute the dominant eigenvector



Inverse iteration



Power iteration

QR algorithm

[Link: Detailed explanation]

QR algorithm

```
def eig(A, n_it=10):
```


[Link: Detailed explanation]

QR algorithm

```python
def eig(A, n_it=10):
    for i in range(n_it):
```


[Link: Detailed explanation]

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
```

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
```


[Link: Detailed explanation]

QR algorithm

```python
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```


[Link: Detailed explanation]

QR algorithm

```python
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

$$\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k$$

[Link: Detailed explanation]

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k\mathbf{Q}_k$$

[Link: Detailed explanation]

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \underline{\mathbf{Q}_k^{-1}\mathbf{Q}_k}\mathbf{R}_k\mathbf{Q}_k$$

[Link: Detailed explanation]

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\underline{\mathbf{Q}_k\mathbf{R}_k}\mathbf{Q}_k$$

[Link: Detailed explanation]

QR algorithm

```python
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{A}_k\mathbf{Q}_k$$

[Link: Detailed explanation]

QR algorithm

```python
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

Similarity transformation: Same eigenvalues

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{A}_k\mathbf{Q}_k$$

QR algorithm

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

Similarity transformation: Same eigenvalues

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{A}_k\mathbf{Q}_k$$

- **Intuition**: Run n instances of the power iteration at the same time.
  - QR factorization renormalizes vectors and keeps the eigenvector estimates from all collapsing onto the same solution.

Demo time

QR algorithm

Similarity transformation: Same eigenvalues

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \underline{\mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k}\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{A}_k\mathbf{Q}_k$$

```
def eig(A, n_it=10):
    for i in range(n_it):
        Q, R = la.qr(A)
        A = R @ Q
    return A
```

- Intuition: Run n instances of the power iteration at the same time.
  - QR factorization renormalizes vectors and keeps the eigenvector estimates from all collapsing onto the same solution.
- **In practice**: many subtleties, use libraries and don't build your own version.

Application: Solving linear ODEs

Simple homogeneous ODE: $x'(t) = ax(t)$

Application: Solving linear ODEs

Simple homogeneous ODE: $x'(t) = ax(t)$

Solutions have the form: $x(t) = e^{at}x(0)$

Application: Solving linear ODEs

Simple homogeneous ODE: $x'(t) = ax(t)$

Solutions have the form: $x(t) = e^{at}x(0)$

let's double-check that: $x'(t) = ae^{at}x(0)$

Application: Solving linear ODEs

Simple homogeneous ODE: $x'(t) = ax(t)$

Solutions have the form: $x(t) = e^{at}x(0)$

let's double-check that: $x'(t) = ae^{at}x(0)$

Same approach also works for systems of ODEs!

System of homogeneous ODEs: $\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t)$

Application: Solving linear ODEs

Simple homogeneous ODE: $x'(t) = ax(t)$

Solutions have the form: $x(t) = e^{at}x(0)$

let's double-check that: $x'(t) = ae^{at}x(0)$

Same approach also works for systems of ODEs!

System of homogeneous ODEs: $\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t)$

Solutions have the form: $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ **when** A **is a matrix.**

?!?!

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

?!?!

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

Standard series definition:

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

?!?!

Standard series definition: $e^a = \sum_{k=0}^{\infty} \frac{1}{k!} a^k$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ **when** A **is a matrix.**

?!?!

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

Standard series definition: $e^{a} = \sum_{k=0}^{\infty} \frac{1}{k!} a^{k}$ $\quad e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^{k}$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

$$e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

$$\begin{aligned} e^{\mathbf{A}} &= \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1} \\ &= \mathbf{V}\left[\sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{\Lambda}^k\right]\mathbf{V}^{-1} \end{aligned}$$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

$$e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$$

$$= \mathbf{V}\left[\sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{\Lambda}^k\right]\mathbf{V}^{-1} = \mathbf{V}e^{\mathbf{\Lambda}}\mathbf{V}^{-1}$$

The Matrix Exponential

A way to evaluate $\exp(\mathrm{A})$ when A is a matrix.

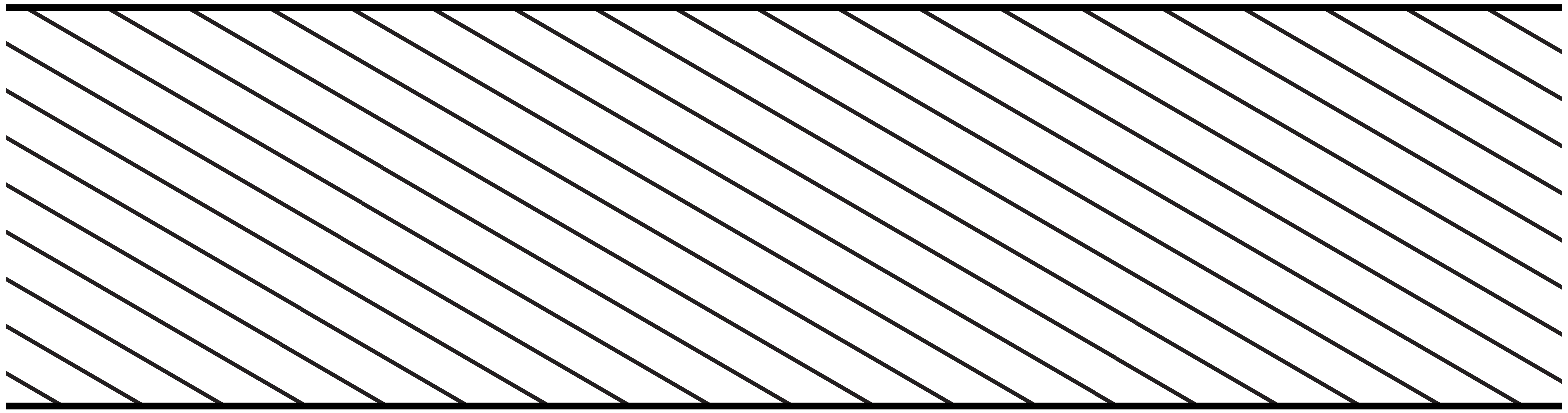
$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0)$$

$$\begin{aligned} e^{\mathbf{A}} &= \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{V}\mathbf{\Lambda}^k \mathbf{V}^{-1} \\ &= \mathbf{V}\left[\sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{\Lambda}^k\right]\mathbf{V}^{-1} = \mathbf{V}e^{\mathbf{\Lambda}}\mathbf{V}^{-1} \end{aligned}$$

$$e^{\mathbf{\Lambda}} = \begin{pmatrix} e^{\lambda_1} & & \\ & \ddots & \\ & & e^{\lambda_n} \end{pmatrix}$$

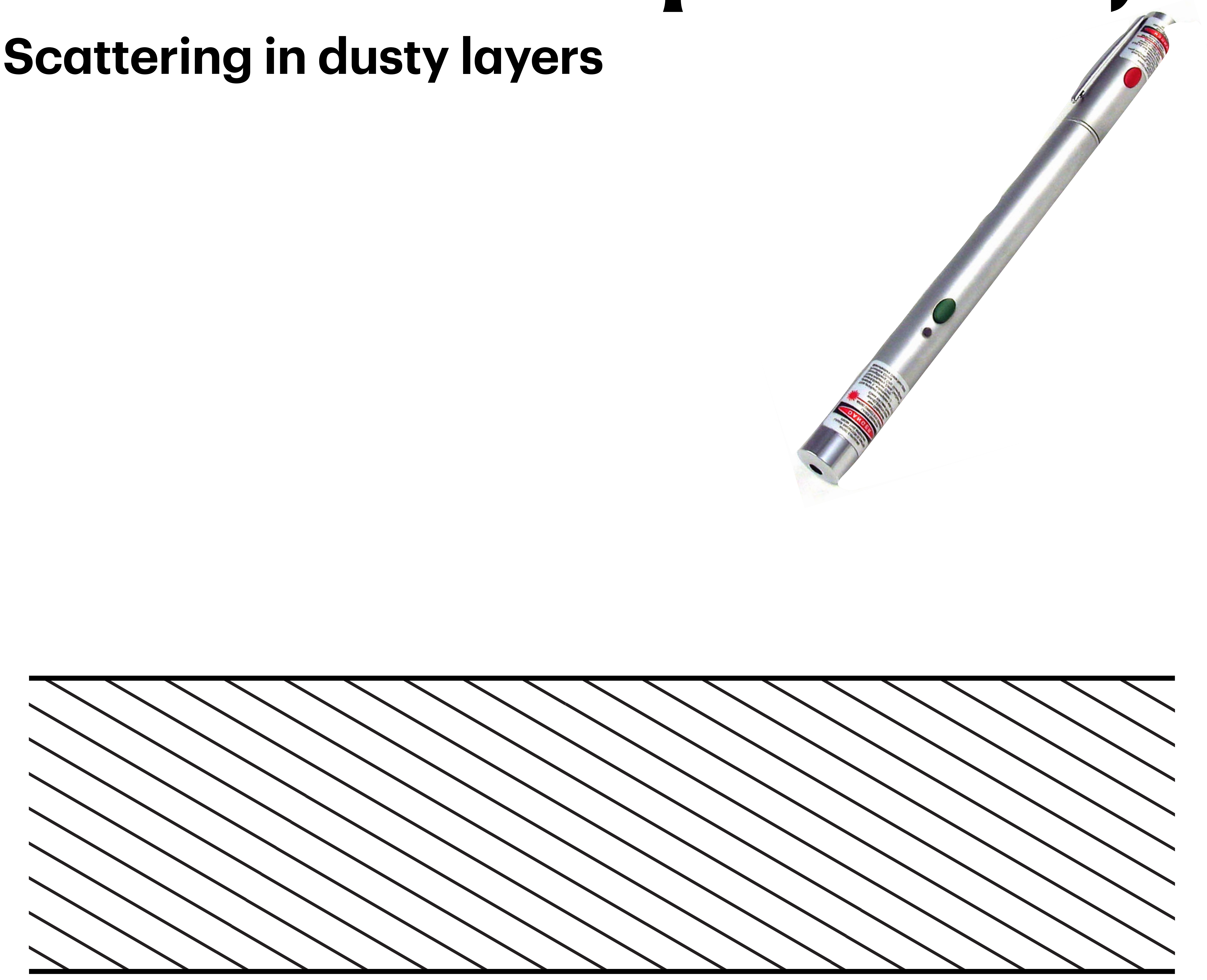
A random example from my own research

Scattering in dusty layers



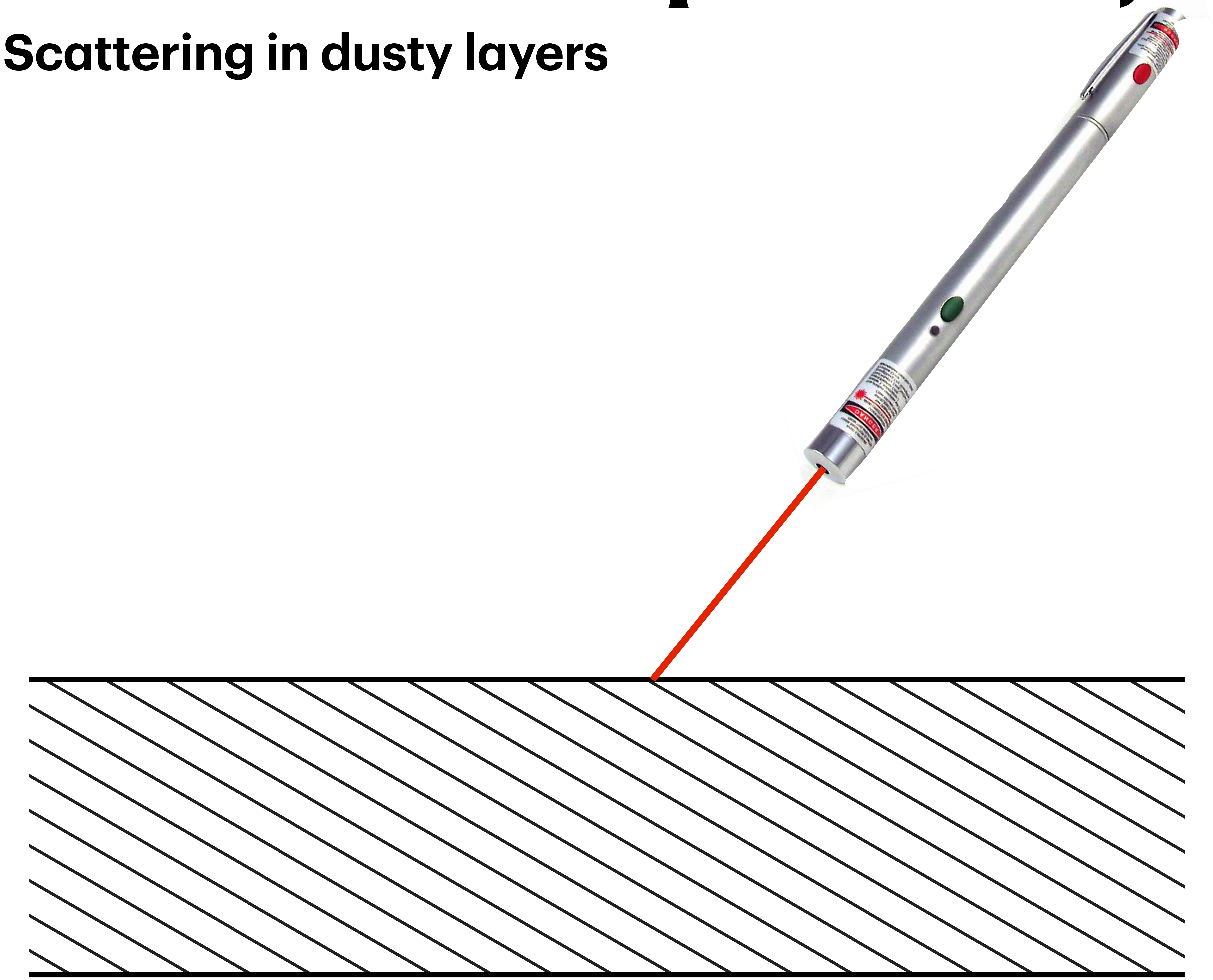
A random example from my own research

Scattering in dusty layers



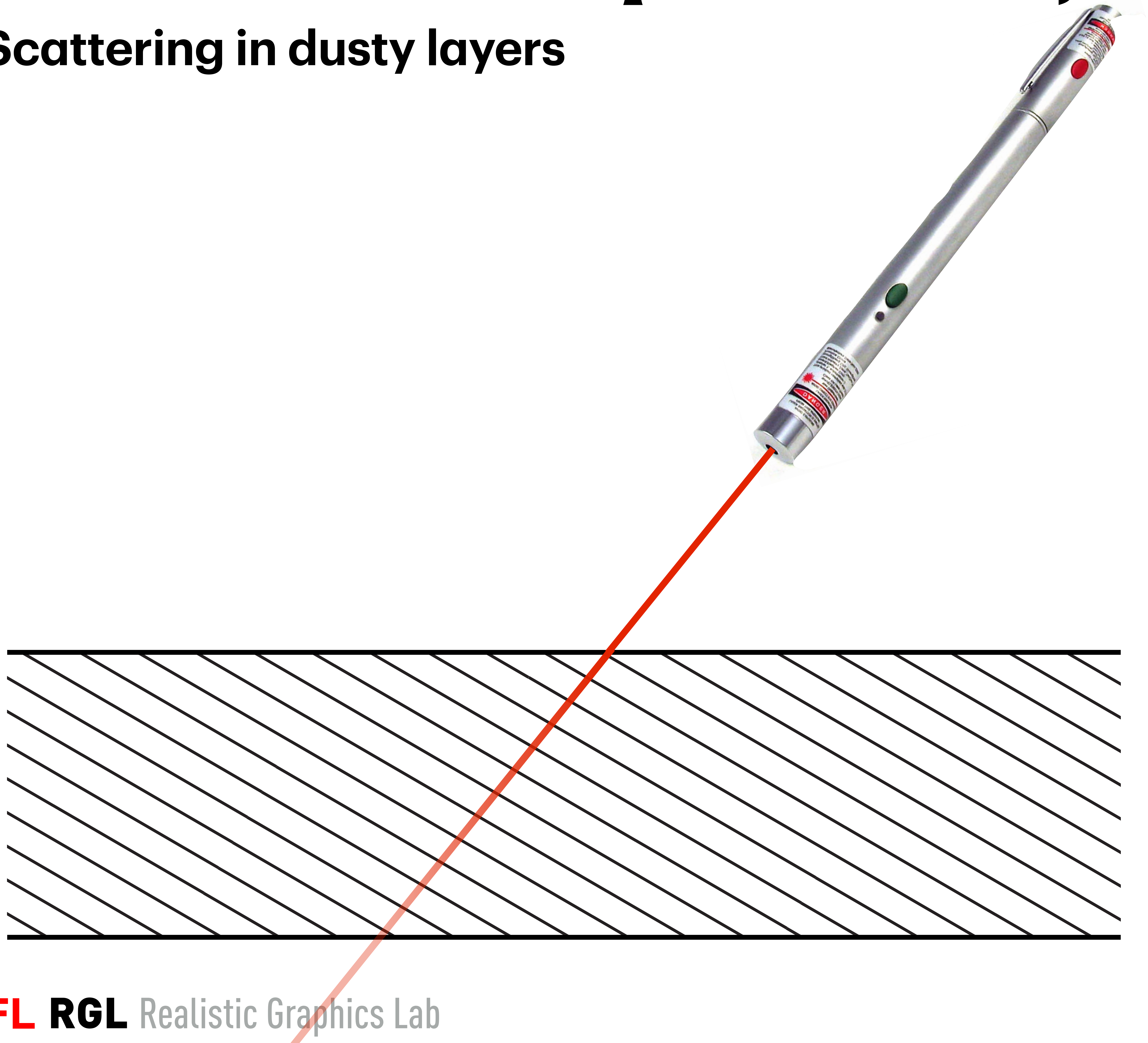
A random example from my own research

Scattering in dusty layers



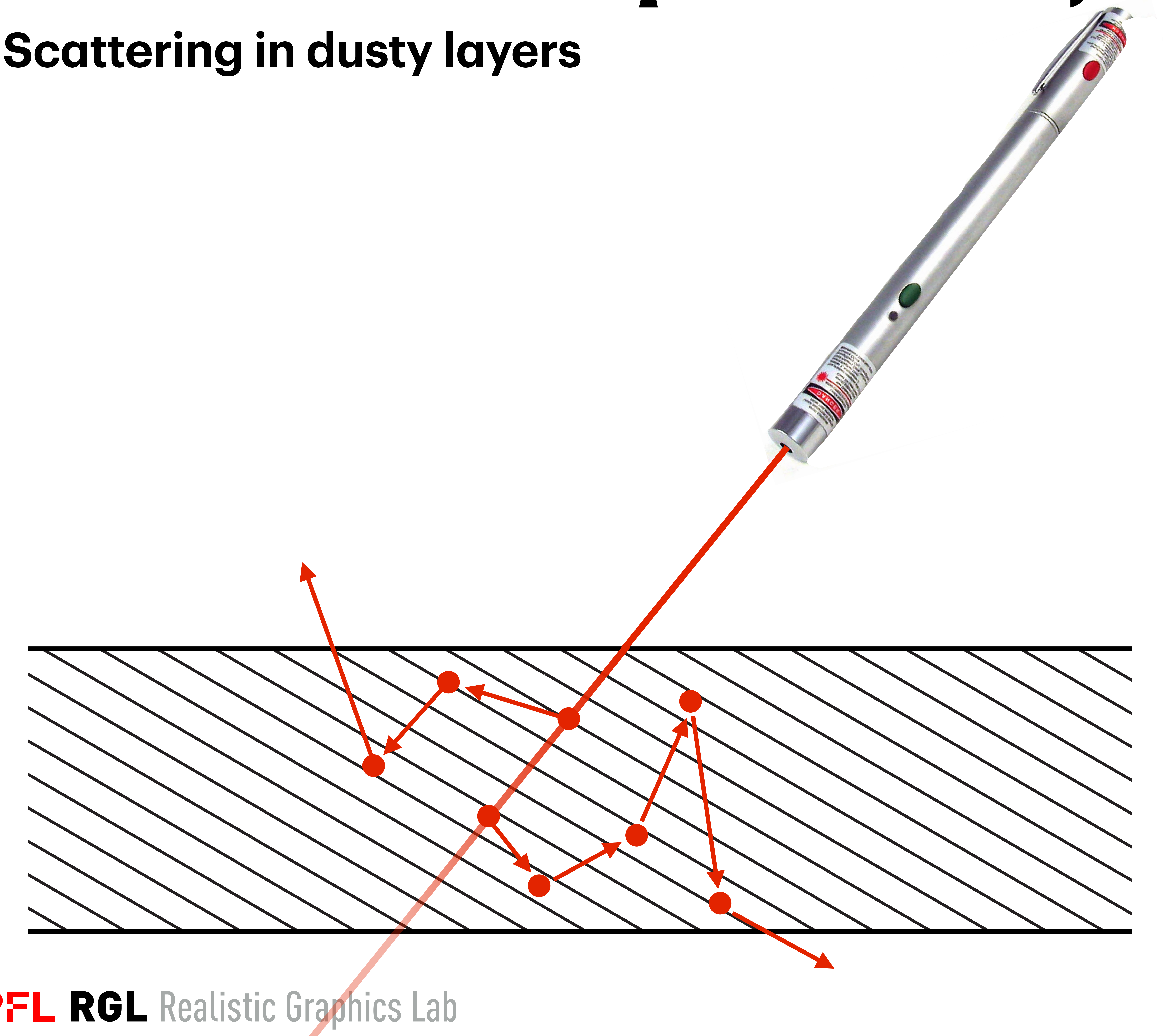
A random example from my own research

Scattering in dusty layers



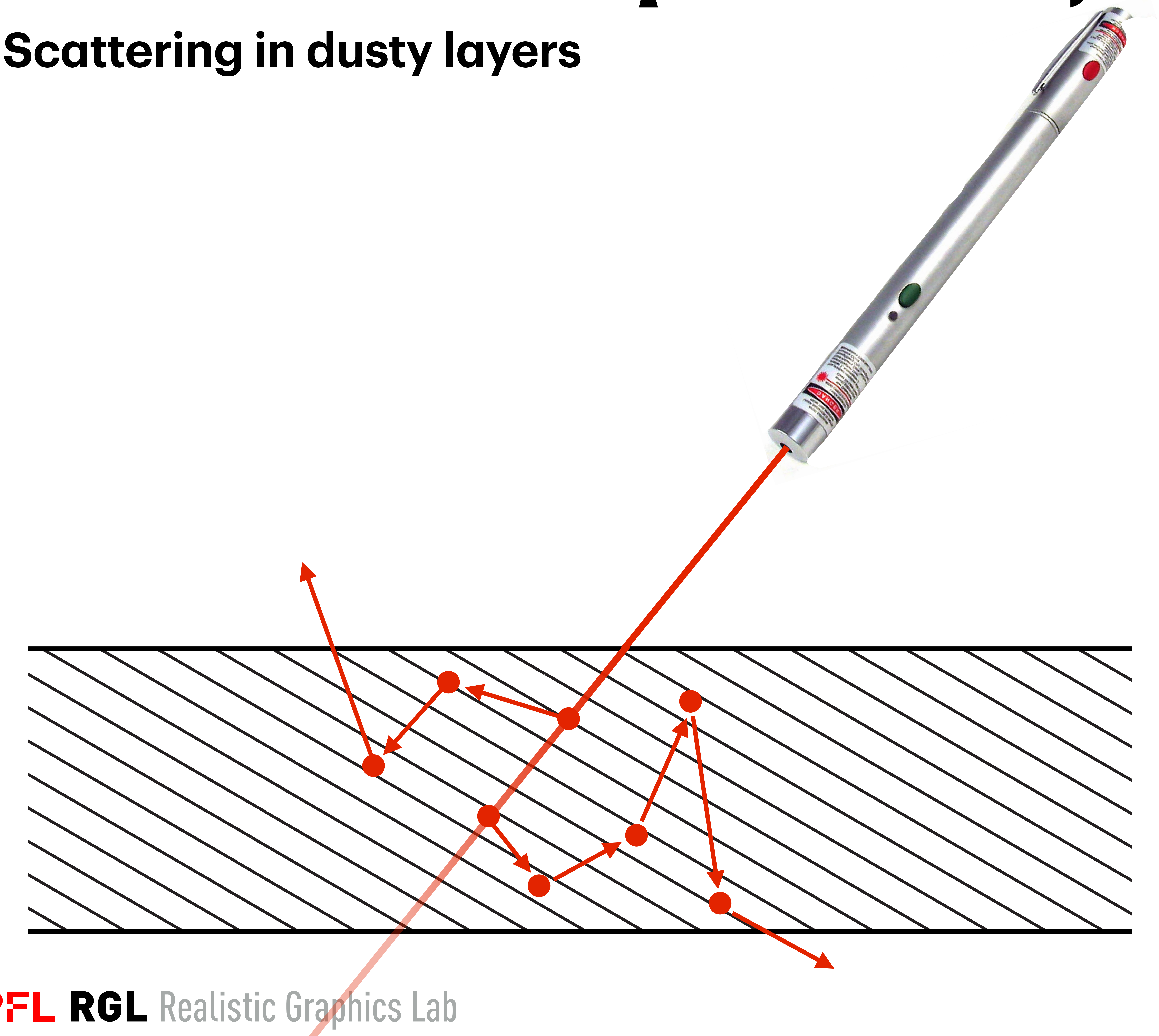
A random example from my own research

Scattering in dusty layers



A random example from my own research

Scattering in dusty layers



Problem has the form

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t)$$

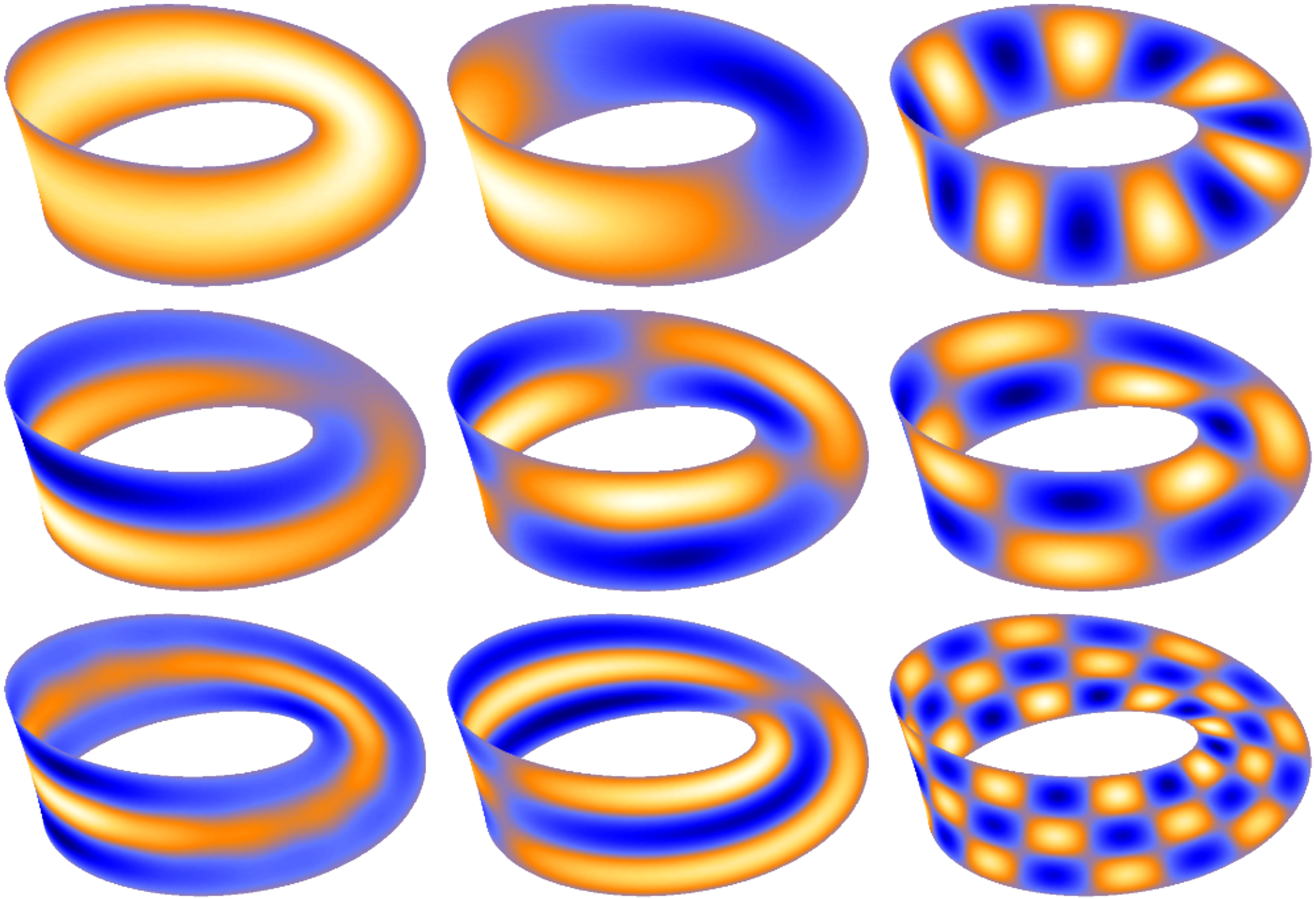
$\mathbf{x}(t)$ is the light traveling in different directions.

t is the depth within the layer.

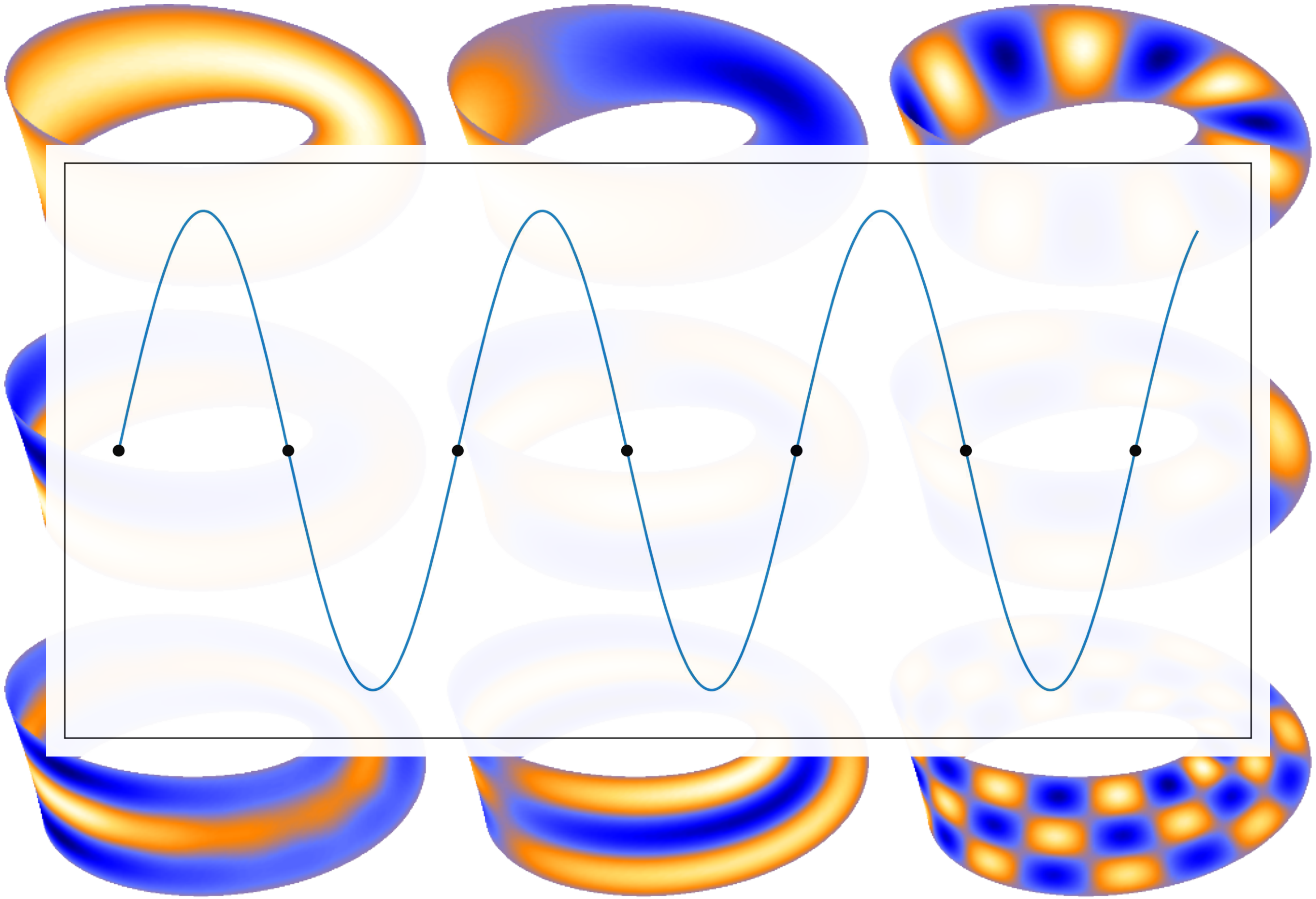

α=0.02
τ=0.000, g=0.900

α=0.02
τ=0.000, g=0.900

Oscillations on surfaces



Oscillations on surfaces



[Langlois, An, Jin, and James, SIGGRAPH 2014]

Diameter: 19 cm
Vertices: 51434
Compressed: 26 kB
Uncompressed: 28 MB

[Langlois, An, Jin, and James, SIGGRAPH 2014]

Diameter: 19 cm
Vertices: 51434
Compressed: 26 kB
Uncompressed: 28 MB

Complex eigenvalues / eigenvectors

An awkward special case

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

Complex eigenvalues / eigenvectors

An awkward special case

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0.$$

Complex eigenvalues / eigenvectors

An awkward special case

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0.$$

$$\Leftrightarrow \quad \lambda^2 - 2\lambda + 2 = 0.$$

Complex eigenvalues / eigenvectors

An awkward special case

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\det\left(\lambda \mathbf{I} - \mathbf{A}\right) = 0.$$

$$\Leftrightarrow \quad \lambda^2 - 2\lambda + 2 = 0.$$

$$\Leftrightarrow \quad \lambda = \frac{2 \pm \sqrt{4-8}}{2} = 1 \pm i.$$

Complex eigenvalues / eigenvectors

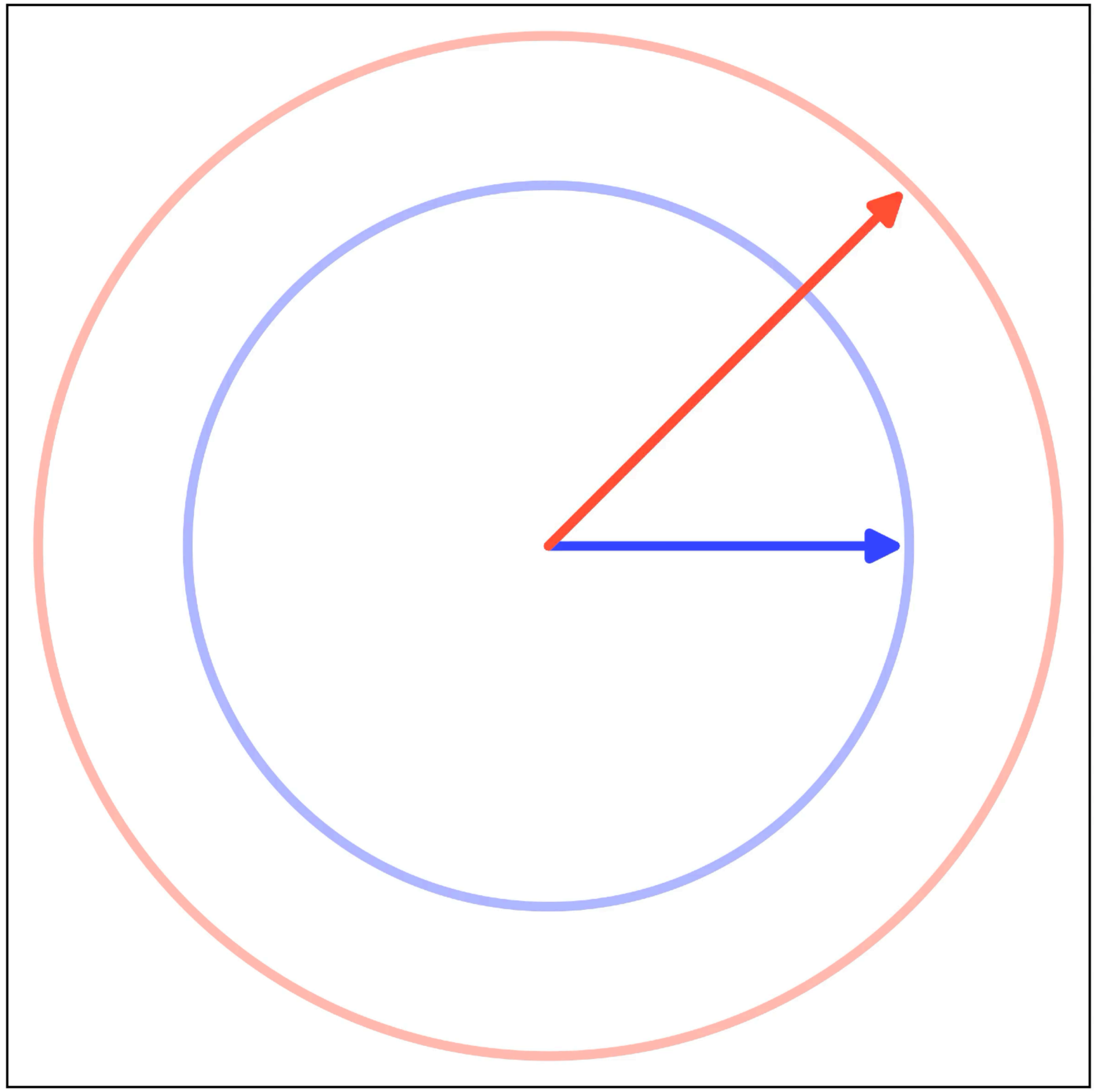
An awkward special case

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0.$$

$$\Leftrightarrow \quad \lambda^2 - 2\lambda + 2 = 0.$$

$$\Leftrightarrow \quad \lambda = \frac{2 \pm \sqrt{4-8}}{2} = 1 \pm i.$$



Eigenanalysis in practice: it's complicated..

1. an eigenvalue/eigenvector "pair" is actually a *triple* $(\mathbf{x}, \mathbf{y}, \lambda)$

where $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$ $\qquad$ $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

"left eigenvector" $\qquad$ *"right eigenvector"*

Eigenanalysis in practice: it's complicated..

1. an eigenvalue/eigenvector "pair" is actually a *triple* $(\mathbf{x}, \mathbf{y}, \lambda)$

where $\mathbf{y}^T \mathbf{A} = \lambda \mathbf{y}^T$ $\quad$ $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$

"left eigenvector" $\quad$ *"right eigenvector"*

2. complex arithmetic generally required.

Eigenanalysis in practice: it's complicated..

non-symmetric case

1. an eigenvalue/eigenvector "pair" is actually a *triple* $(\mathbf{x}, \mathbf{y}, \lambda)$

where $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$ *"left eigenvector"* $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ *"right eigenvector"*

2. complex arithmetic generally required.

Eigenanalysis in practice: it's complicated..

non-symmetric case

1. an eigenvalue/eigenvector "pair" is actually a *triple* $(\mathbf{x}, \mathbf{y}, \lambda)$

where $\mathbf{y}^T \mathbf{A} = \lambda \mathbf{y}^T$ — *"left eigenvector"* — $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$ — *"right eigenvector"*

2. complex arithmetic generally required.

3. methods can be very unstable if eigenvalues not well-separated.

Eigenanalysis in practice: it's complicated..

non-symmetric case

1. an eigenvalue/eigenvector "pair" is actually a *triple* $(\mathbf{x}, \mathbf{y}, \lambda)$

where $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$ $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

"left eigenvector" *"right eigenvector"*

2. complex arithmetic generally required.

3. methods can be very unstable if eigenvalues not well-separated.

4. Doesn't make sense for non-square **A**!

The Singular Value Decomposition

LU $$A = LU$$

QR $$A = QR$$

SVD $$A = U \Sigma V^T$$

The Singular Value Decomposition

The Singular Value Decomposition

- Can be computed for **any** matrix (non-square, non-symmetric, singular, ..)
- Involves only orthogonal and diagonal matrices.
- **Impeccable** numerical properties.
- No complex arithmetic necessary.
- Expensive to compute (~5-10 times the cost of LU)
- SVD and Eigendecomposition are *identical** when **A** is symmetric.

* Except for tiny differences: ordering of entries, etc.

$$A = U \Sigma V^T$$

Terminology

Let's look at the m=n case first.

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

"*Right singular vectors*"
Orthogonal.

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

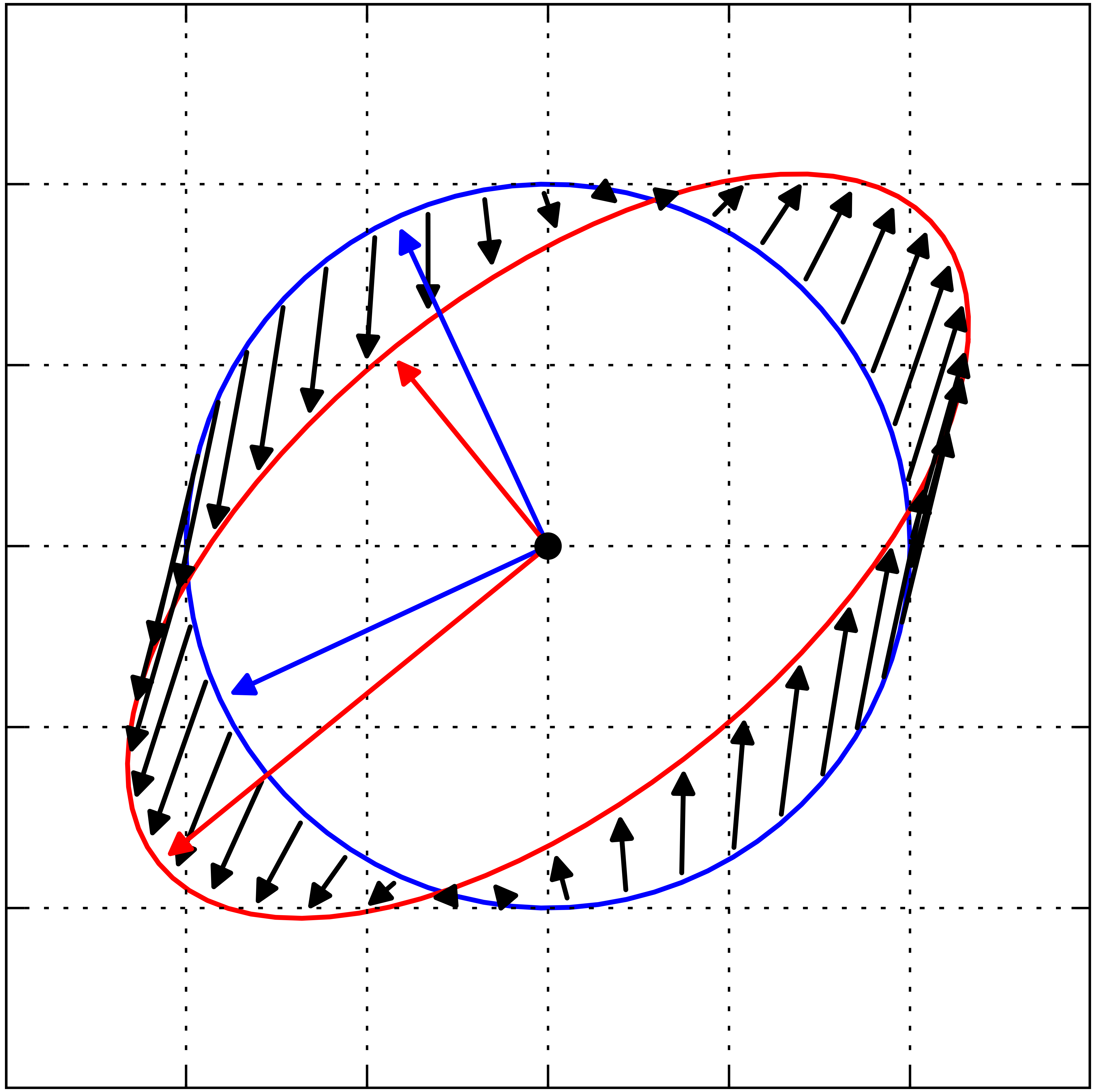
"*Left singular vectors*"
Orthogonal.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}$$

"*Singular values*"
Positive, sorted in decreasing order

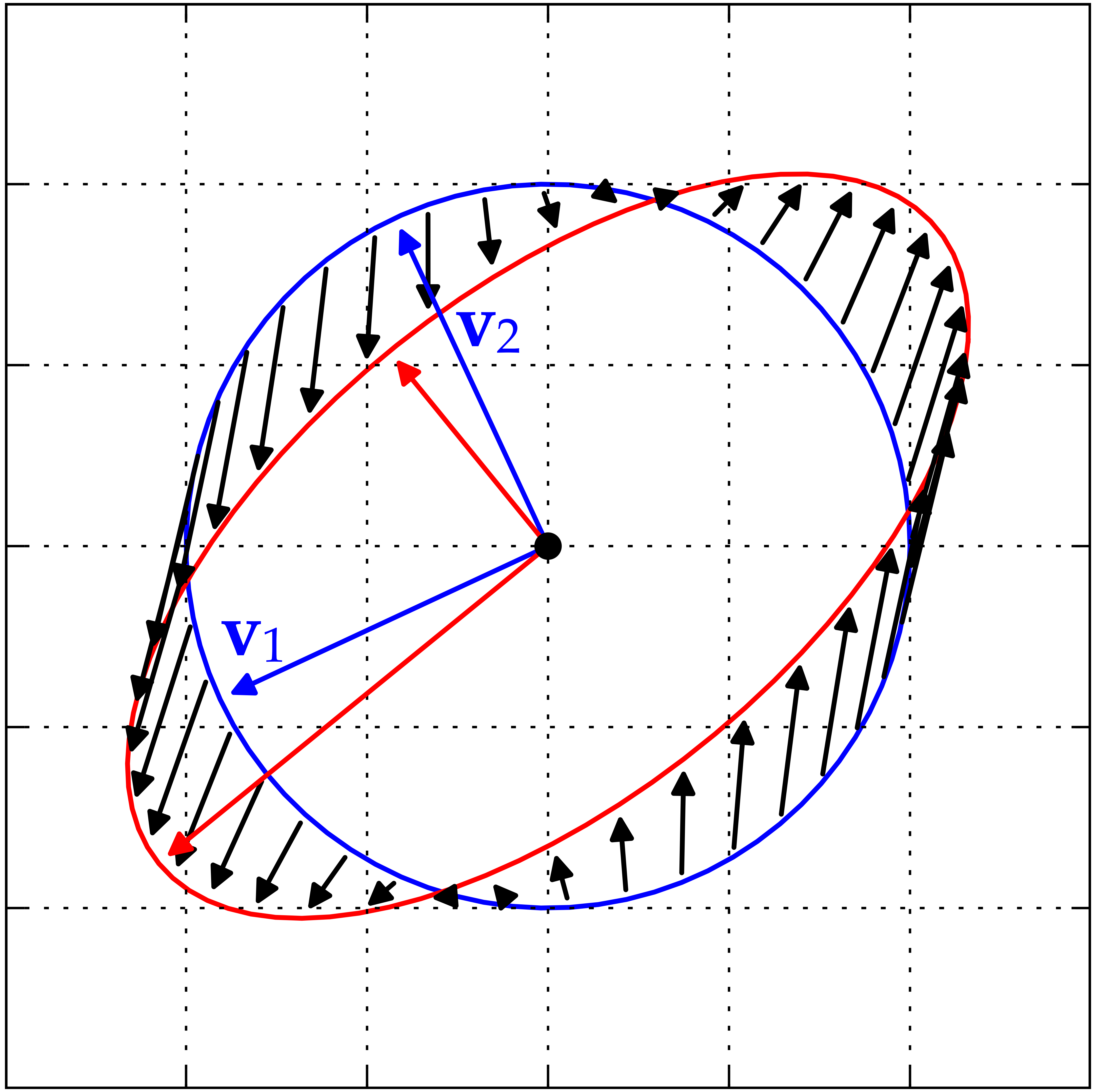
SVD

```
U, Σ, V = scipy.linalg.svd(A)
```



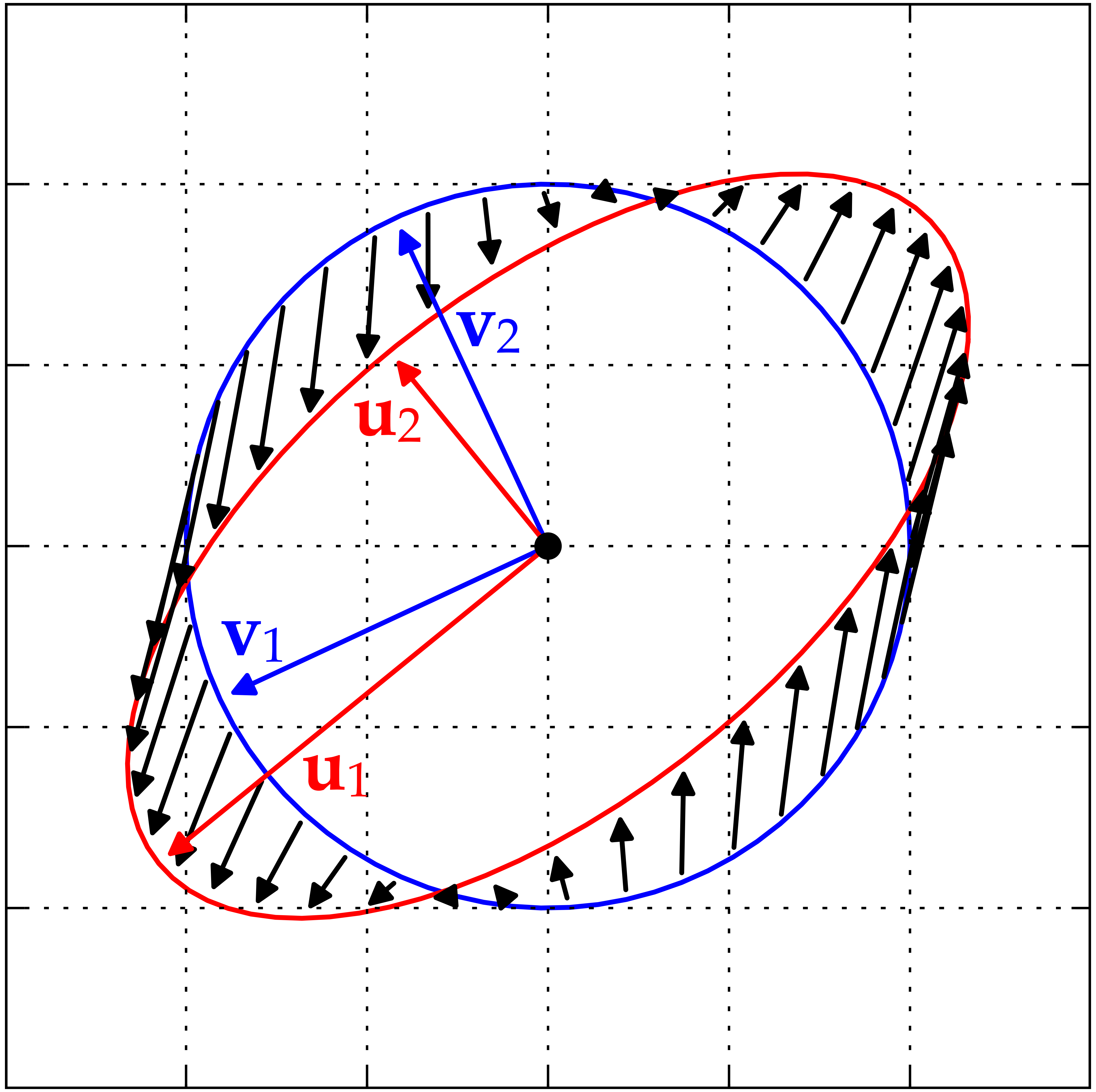
SVD

```
U, Σ, V = scipy.linalg.svd(A)
```



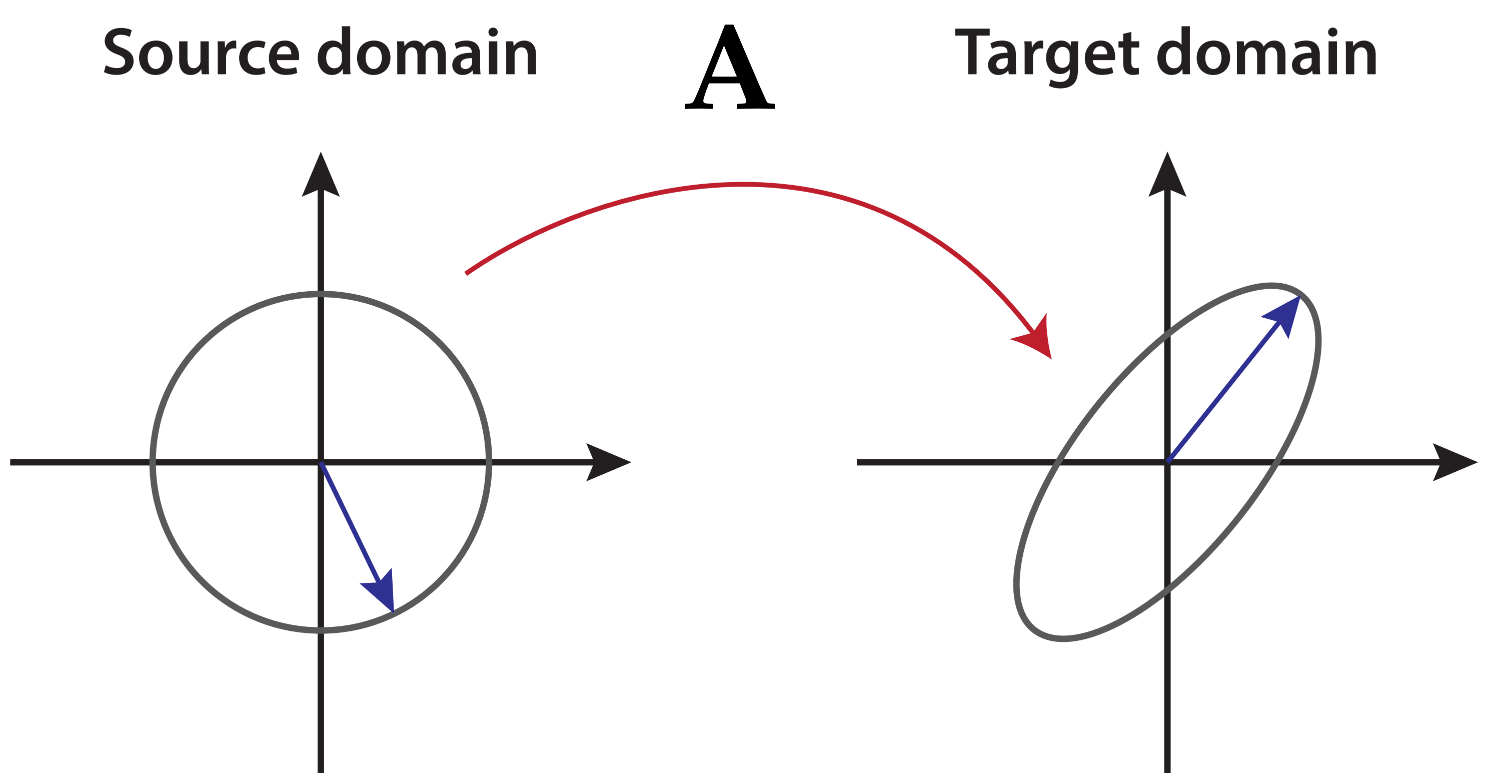
SVD

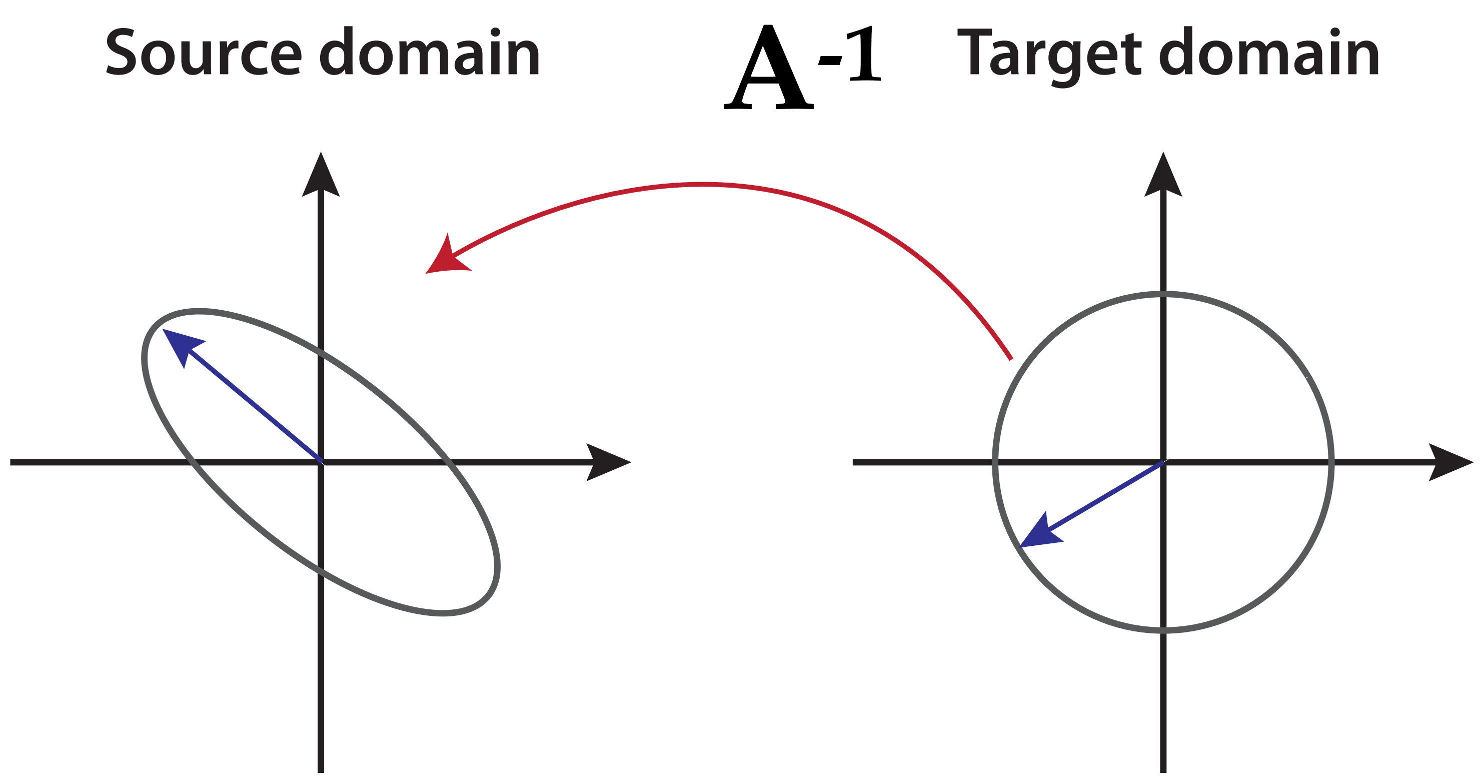
```
U, Σ, V = scipy.linalg.svd(A)
```



Revisiting the condition number

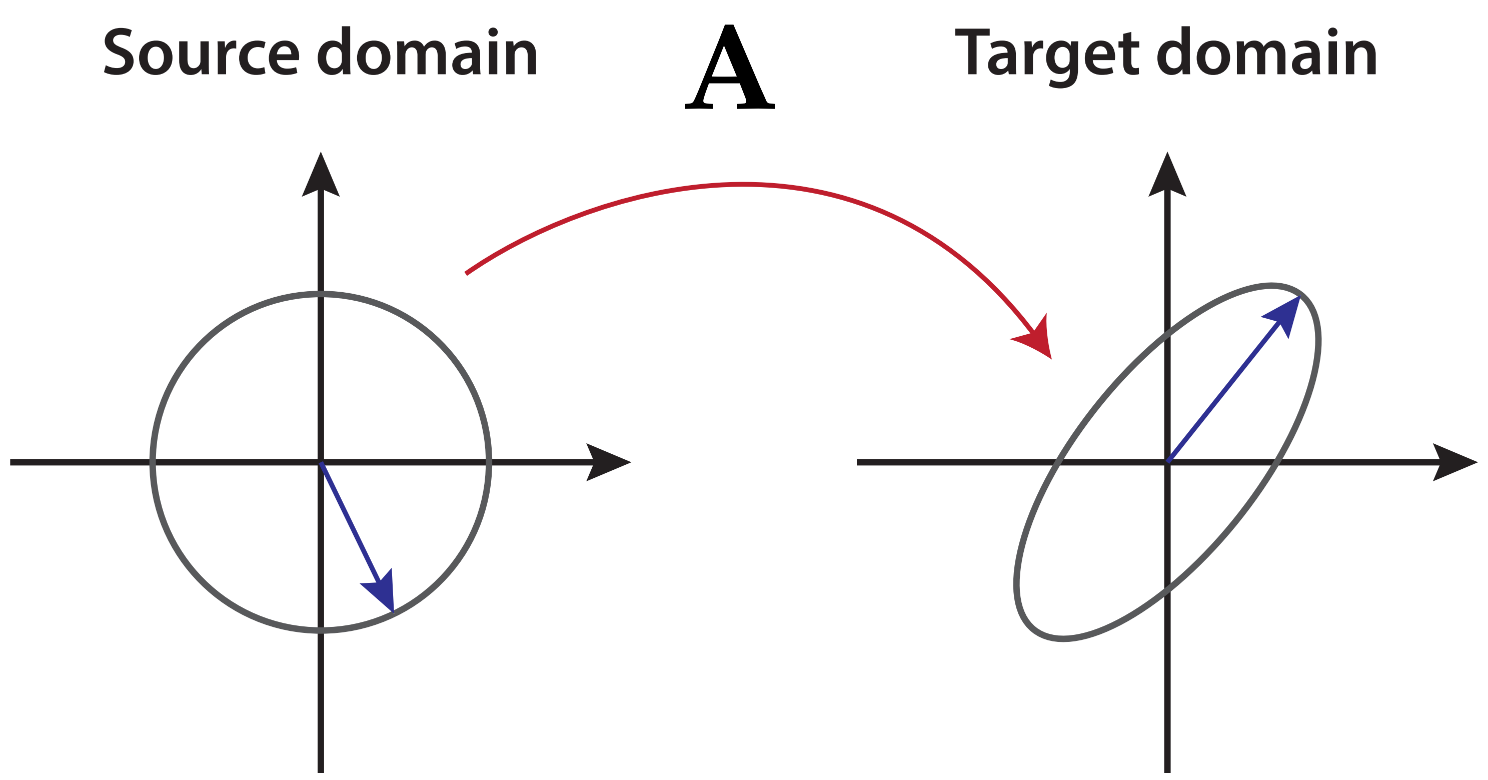
$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

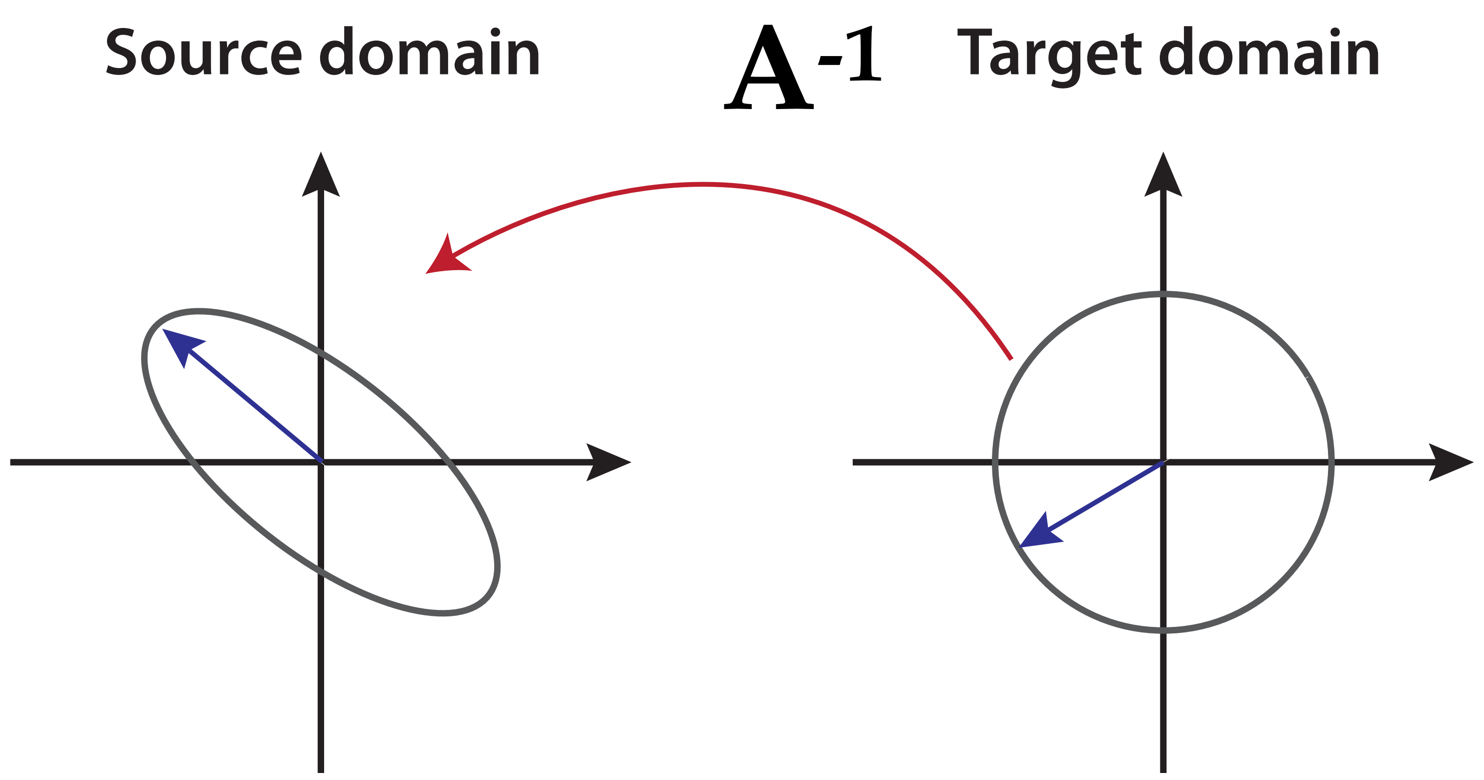




Revisiting the condition number

$$\operatorname{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$
$$= \frac{\sigma_1}{\sigma_n}.$$





Another view of the SVD

$$\begin{bmatrix} \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \end{bmatrix} \begin{bmatrix} \diagdown \mathbf{\Sigma} \diagdown \end{bmatrix} \begin{bmatrix} \mathbf{V}^T \end{bmatrix}$$

Another view of the SVD

$$\begin{bmatrix} \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \end{bmatrix} \begin{bmatrix} \diagdown \boldsymbol{\Sigma} \diagdown \end{bmatrix} \begin{bmatrix} \mathbf{V}^T \end{bmatrix}$$

where

$$\mathbf{V} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}$$

Another view of the SVD

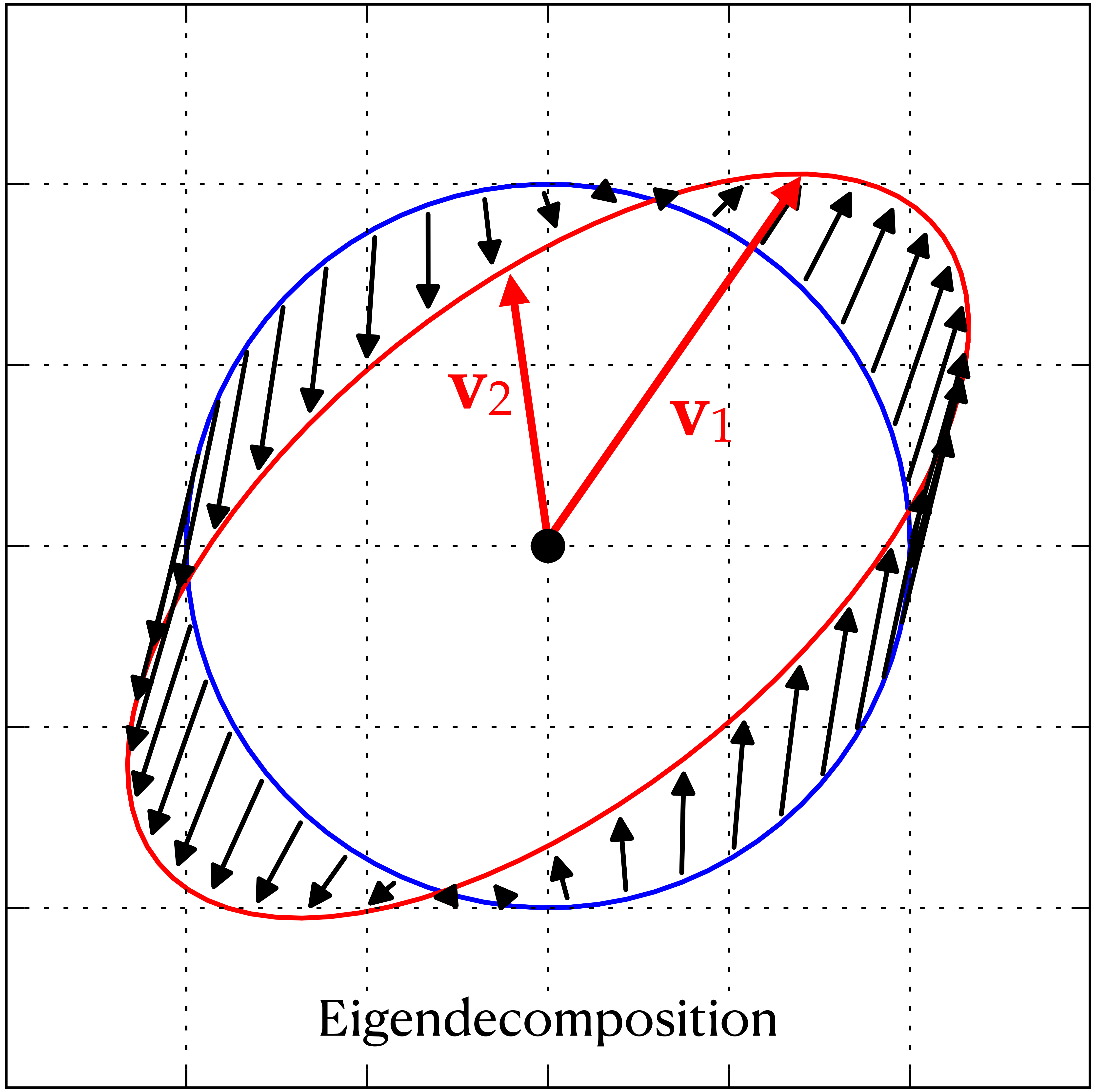
$$\left[\mathbf{A}\right] = \left[\mathbf{U}\right]\left[\diagdown\boldsymbol{\Sigma}\diagdown\right]\left[\mathbf{V}^T\right]$$

where

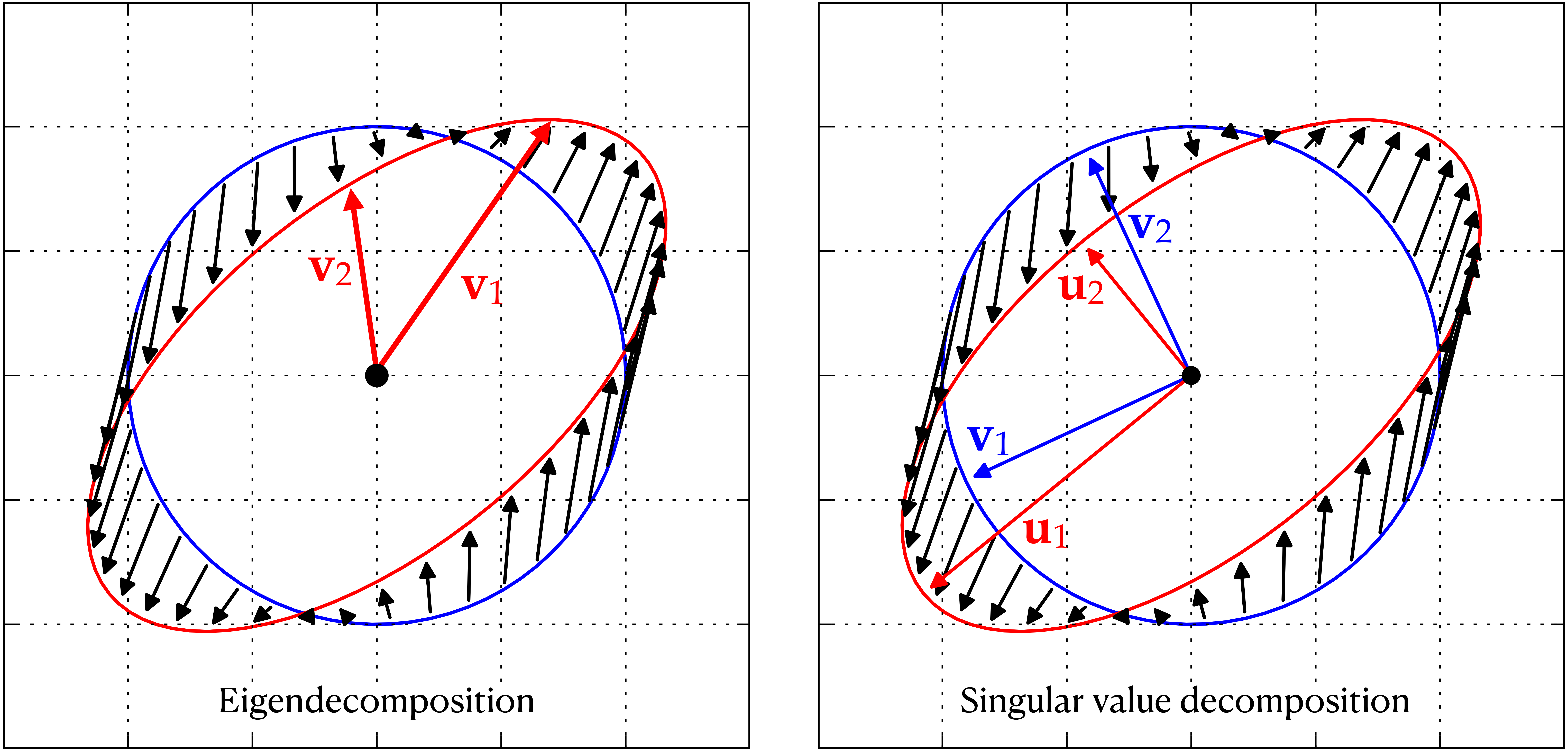
$$\mathbf{V} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}$$

Then $\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$

Eigendecomposition vs. SVD



Eigendecomposition vs. SVD



MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2 \right)$$

normalization constant

exponent

MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

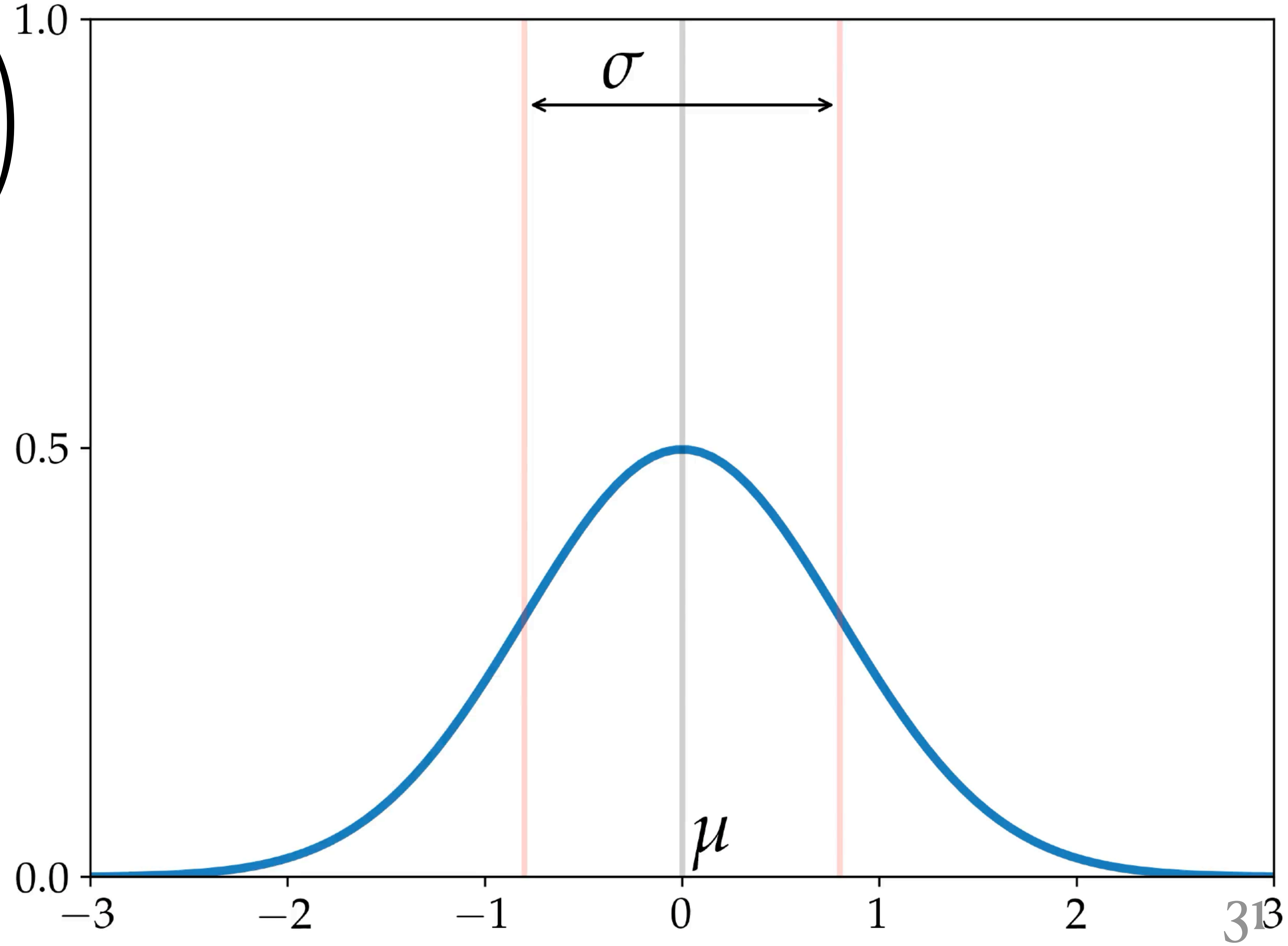
$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

normalization constant

exponent



MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

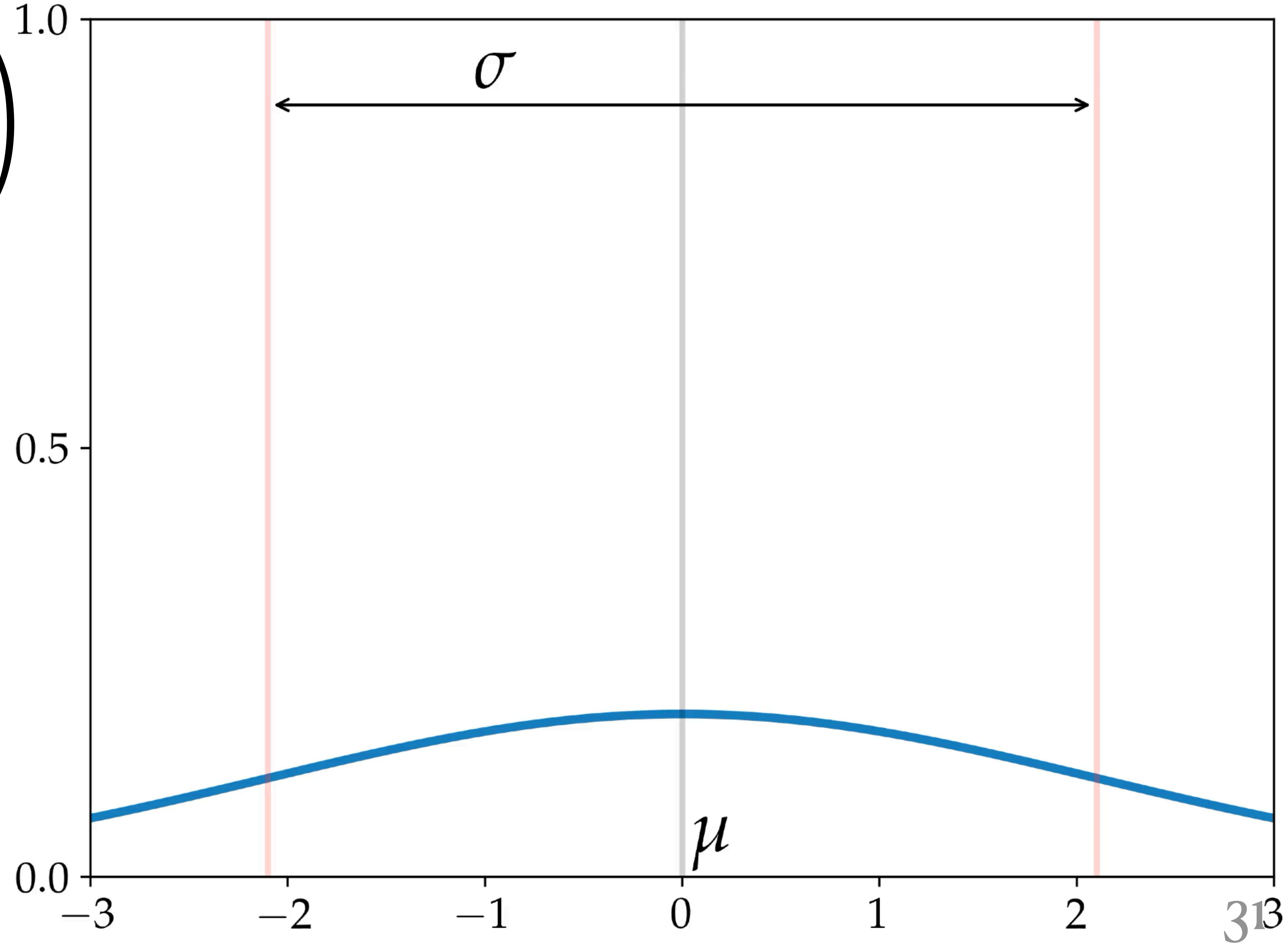
$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

normalization constant

exponent



MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

normalization constant

exponent

MATH-232 Review: The Normal Distribution

Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Variance (square of std. deviation)

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

normalization constant

exponent

Standard deviation

MATH-232 Review: The Normal Distribution

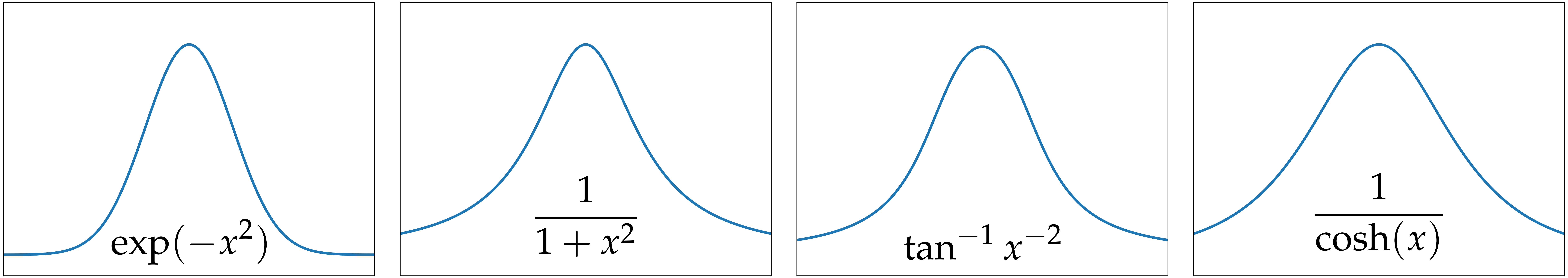
Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2 \right)$$

normalization constant   exponent



MATH-232 Review: The Normal Distribution

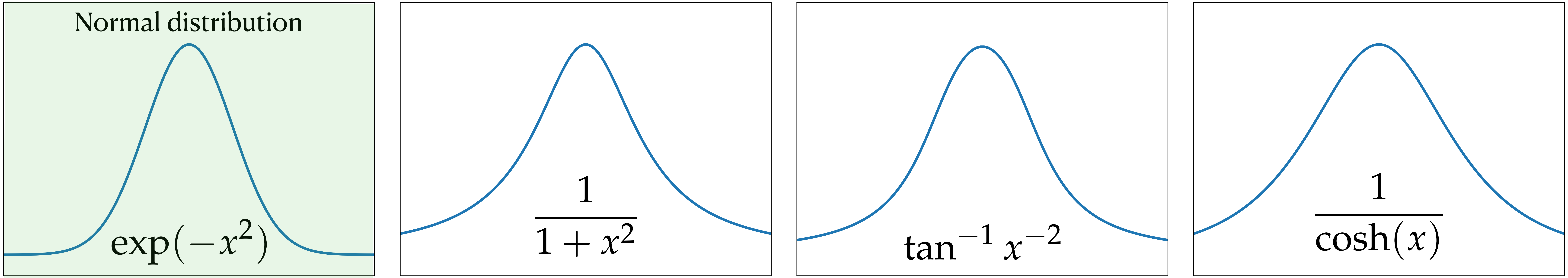
Univariate (1D) case. Also known as "*Gaussian distribution*"

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Density function:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

normalization constant   exponent



The Normal Distribution

Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

The Normal Distribution

Why is *this particular* function interesting?

$X_1, X_2, \ldots$ iid. random variables with $E[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

The Normal Distribution

Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem

The Normal Distribution

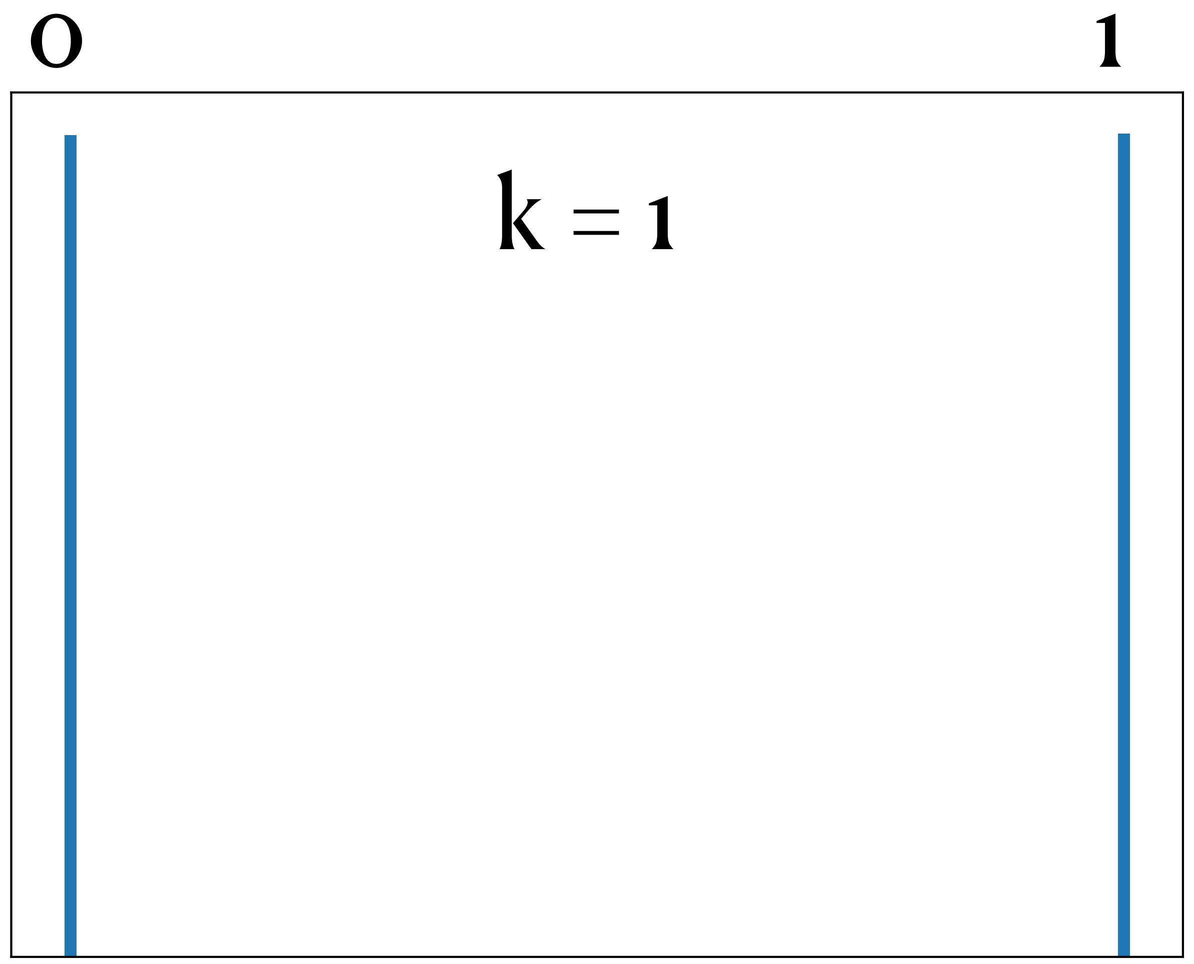
Why is *this particular* function interesting?

$X_1, X_2, \ldots$ iid. random variables with $E[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}\left(\bar{X}_k - \mu\right) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



The Normal Distribution

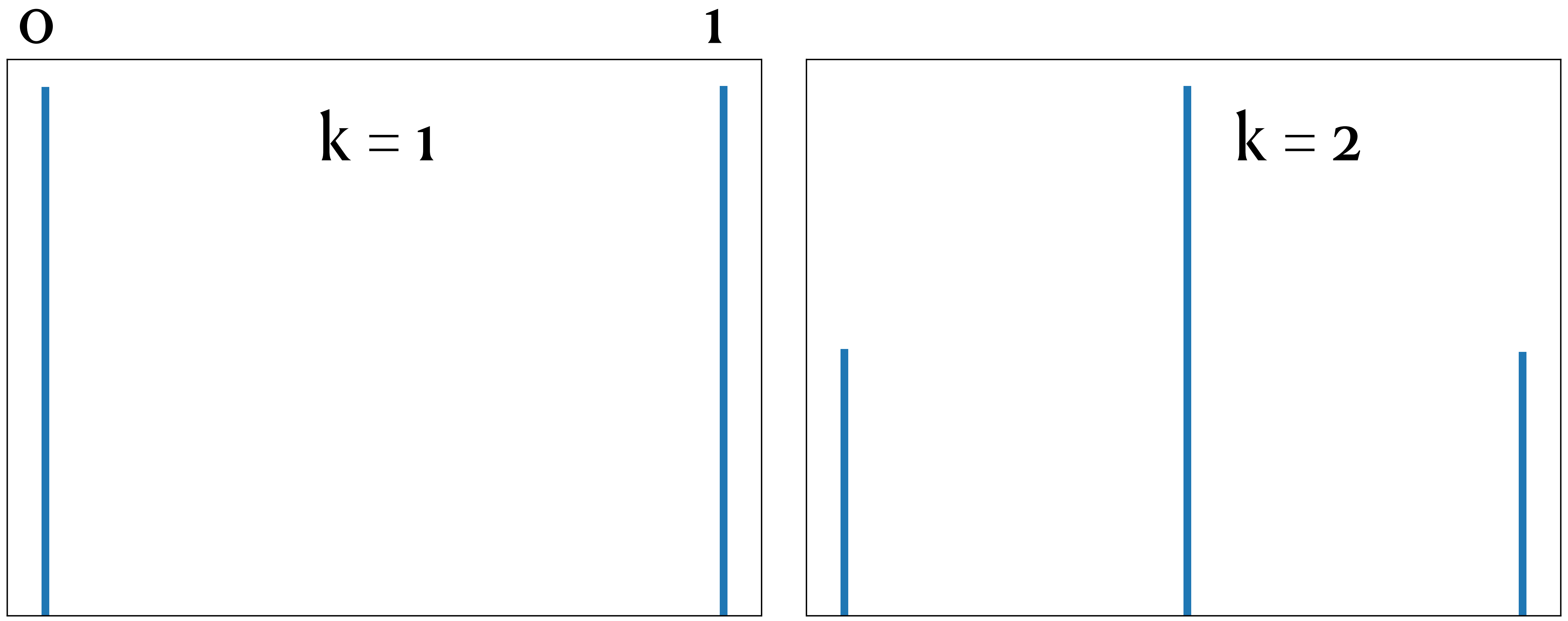
Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}\left(\bar{X}_k - \mu\right) \overset{\text{i.d.}}{\to} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



The Normal Distribution

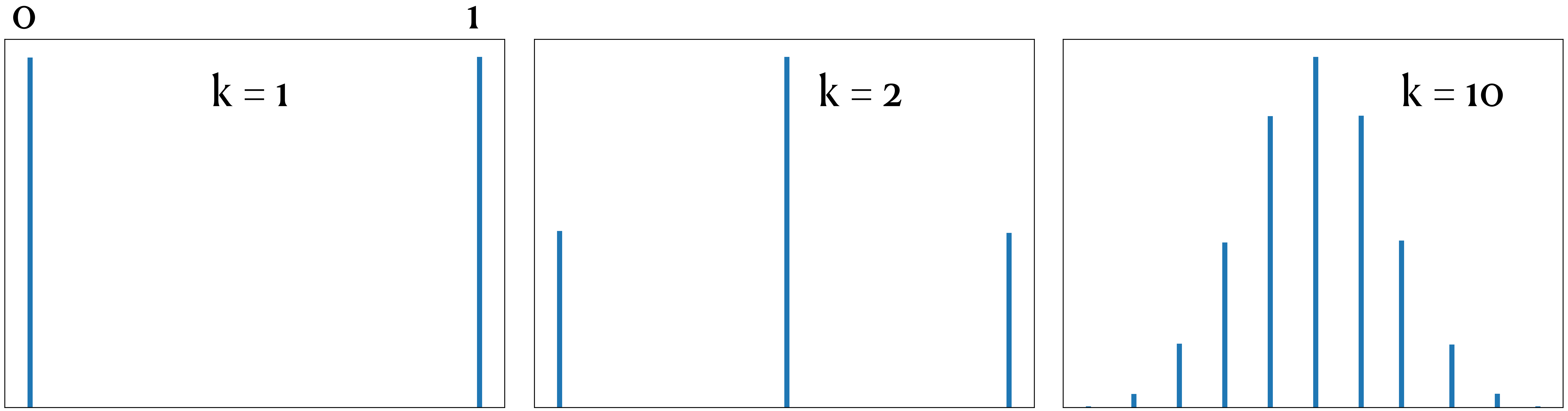
Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



The Normal Distribution

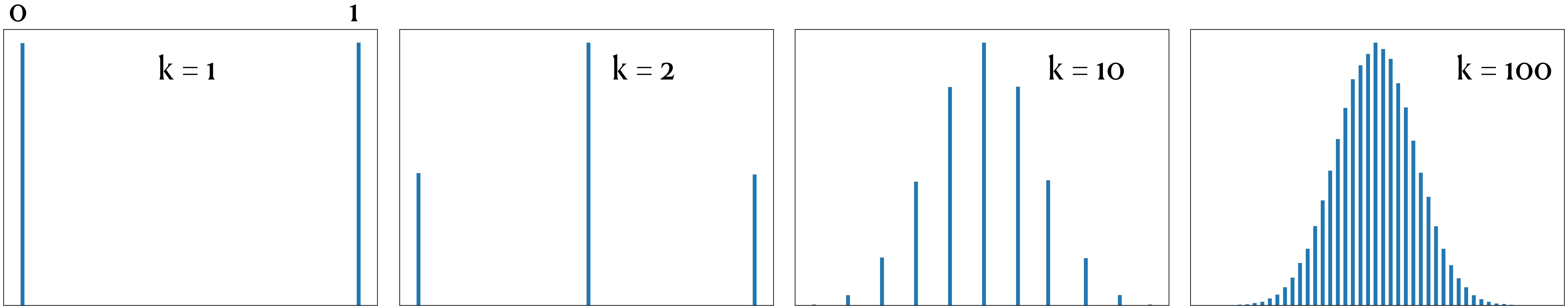
Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



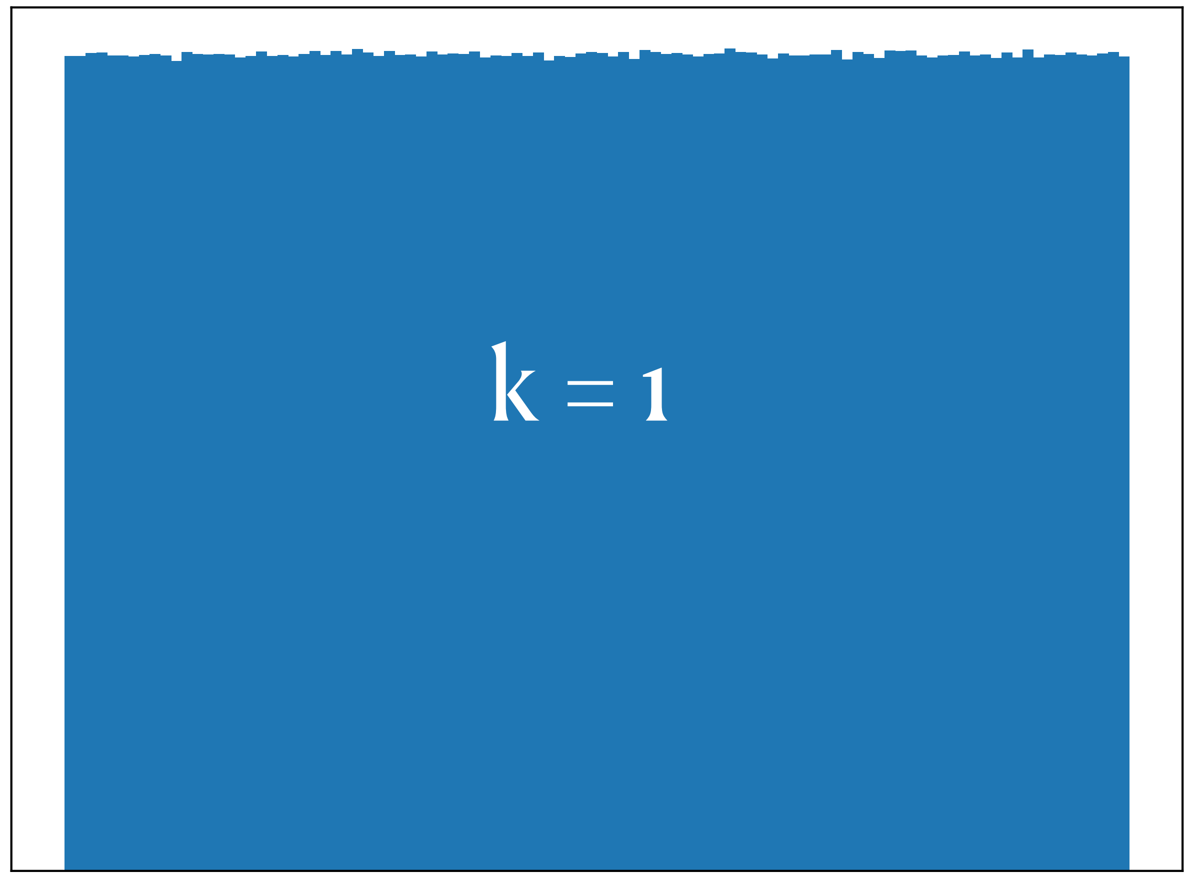
The Normal Distribution

Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$ *Central Limit Theorem*



The Normal Distribution

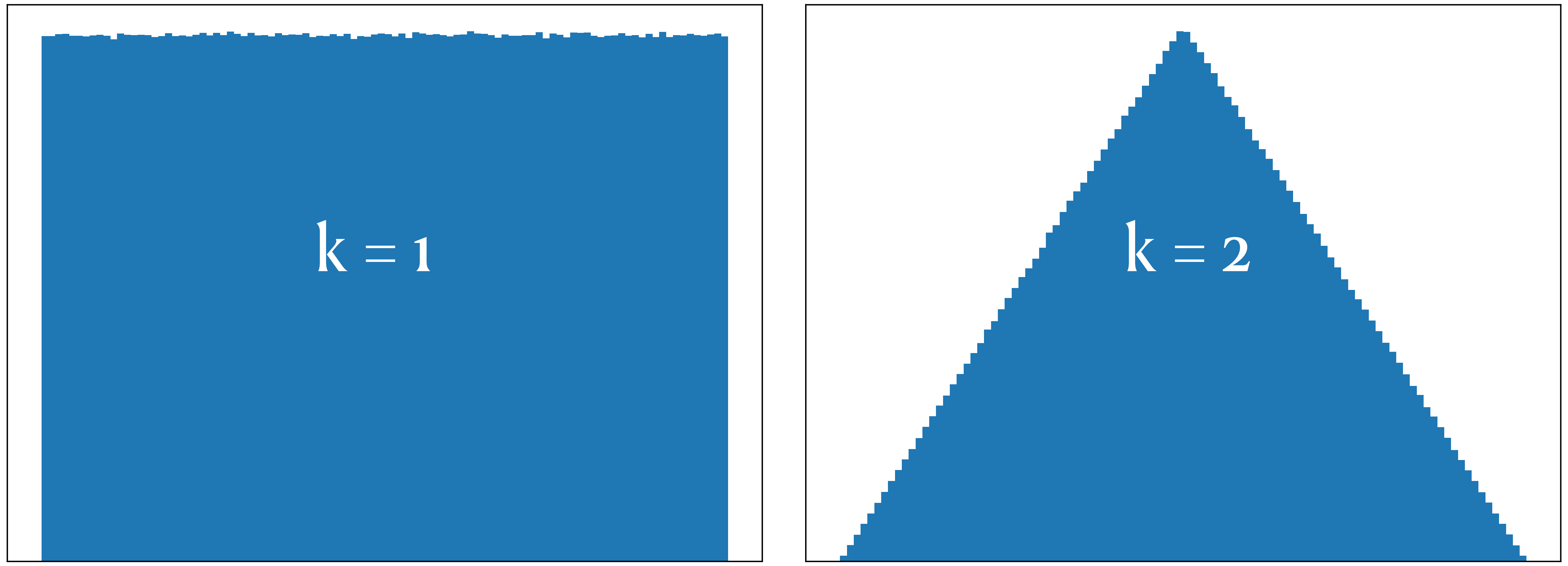
Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



The Normal Distribution

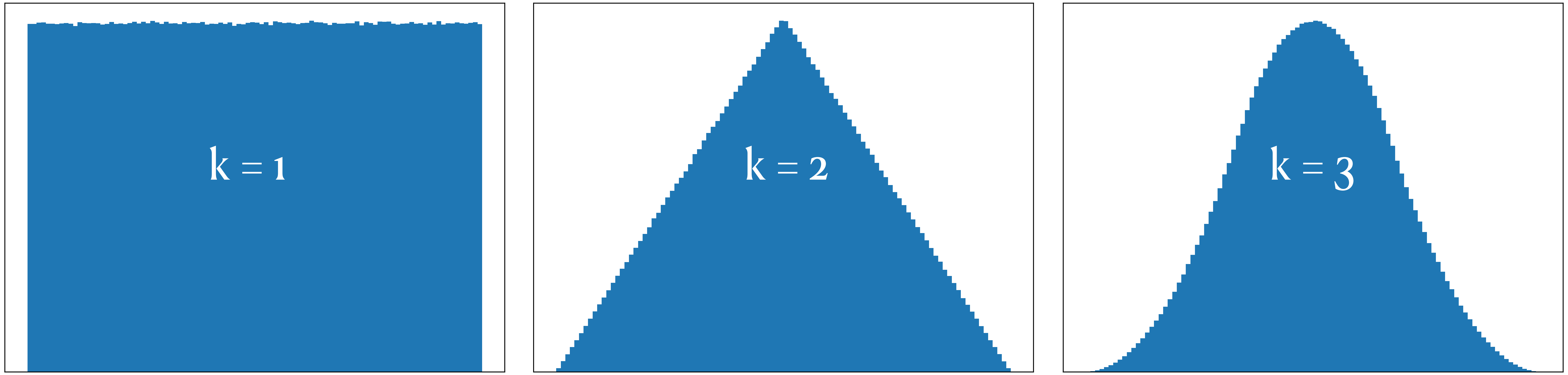
Why is *this particular* function interesting?

$$X_1, X_2, \ldots \quad \text{iid. random variables with } E[X_i] = \mu \text{ and } \mathrm{Var}[X_i] = \sigma^2$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}(\bar{X}_k - \mu) \overset{\text{i.d.}}{\to} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



The Normal Distribution

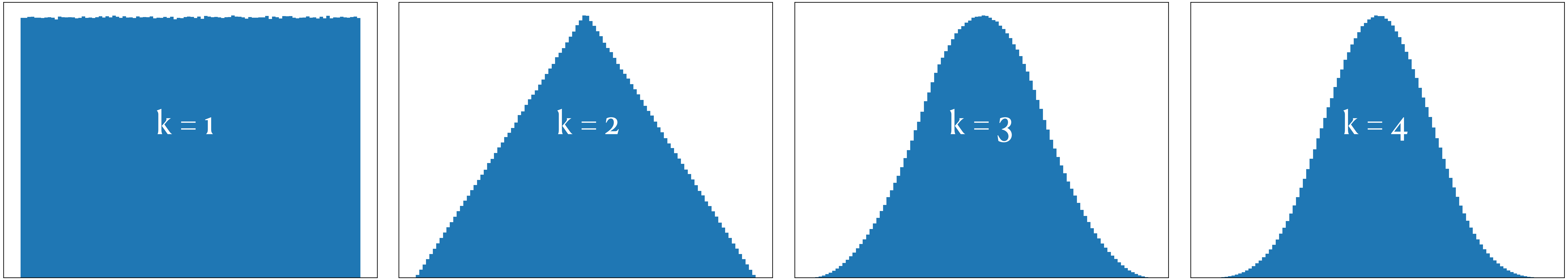
Why is *this particular* function interesting?

$X_1, X_2, \ldots$ iid. random variables with $E[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\frac{\sqrt{k}}{\sigma}\left(\bar{X}_k - \mu\right) \xrightarrow{\text{i.d.}} \mathcal{N}(0,1) \quad (k \to \infty)$$

Central Limit Theorem



Analysing data in higher dimensions (2D)

$$X_1, X_2, \ldots$$

Analysing data in higher dimensions (2D)

$X_1, X_2, \ldots$ $Y_1, Y_2, \ldots$

Analysing data in higher dimensions (2D)

$$X_1, X_2, \ldots$$

$$Y_1, Y_2, \ldots$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k}$$

$$\bar{Y}_k = \frac{Y_1 + \ldots + Y_k}{k}$$

Analysing data in higher dimensions (2D)

$$X_1, X_2, \ldots \qquad\qquad Y_1, Y_2, \ldots$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k} \qquad\qquad \bar{Y}_k = \frac{Y_1 + \ldots + Y_k}{k}$$

$\bar{X}_k \to \mu_X$ (averages converge to mean) $\bar{Y}_k \to \mu_Y$

Analysing data in higher dimensions (2D)

$$X_1, X_2, \ldots \qquad\qquad Y_1, Y_2, \ldots$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k} \qquad\qquad \bar{Y}_k = \frac{Y_1 + \ldots + Y_k}{k}$$

$$\bar{X}_k \to \mu_X \qquad \text{(averages converge to mean)} \qquad \bar{Y}_k \to \mu_Y$$

$$\overline{(X_k - \bar{X}_k)^2} \to \sigma_X^2 \qquad \text{(squared deviations converge to variance)} \qquad \overline{(Y_k - \bar{Y}_k)^2} \to \sigma_Y^2$$

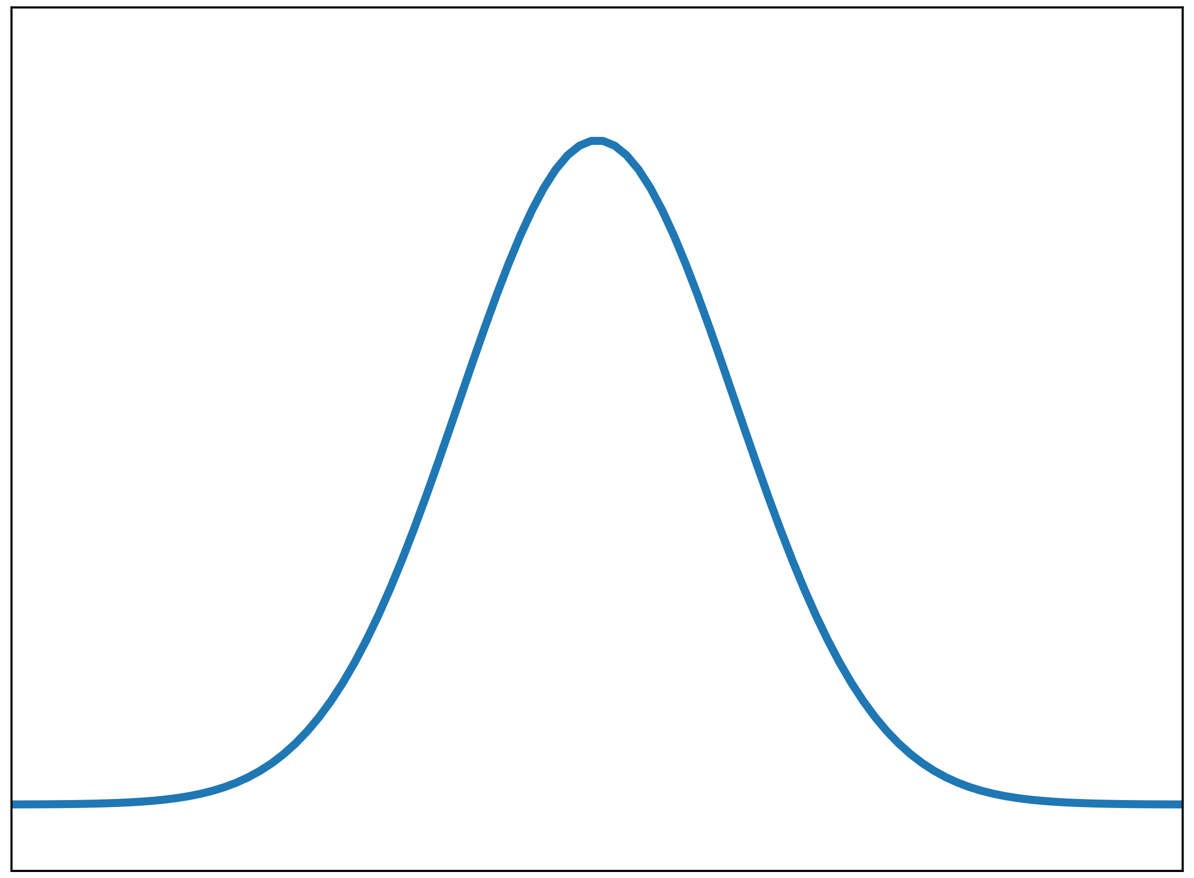
Analysing data in higher dimensions (2D)

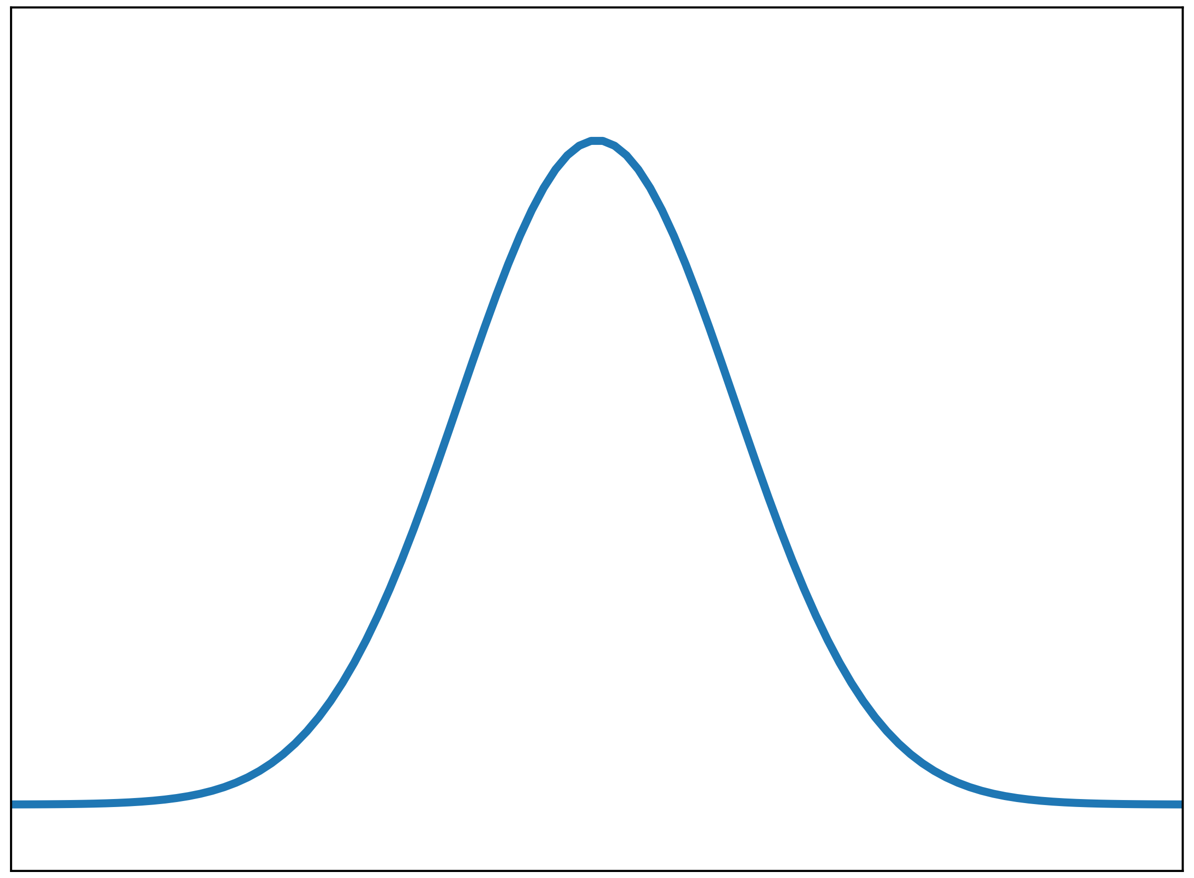
$$X_1, X_2, \ldots \qquad\qquad Y_1, Y_2, \ldots$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k} \qquad\qquad \bar{Y}_k = \frac{Y_1 + \ldots + Y_k}{k}$$

$$\bar{X}_k \to \mu_X$$ (averages converge to mean) $$\bar{Y}_k \to \mu_Y$$

$$\overline{(X_k - \bar{X}_k)^2} \to \sigma_X^2$$ (squared deviations converge to variance) $$\overline{(Y_k - \bar{Y}_k)^2} \to \sigma_Y^2$$





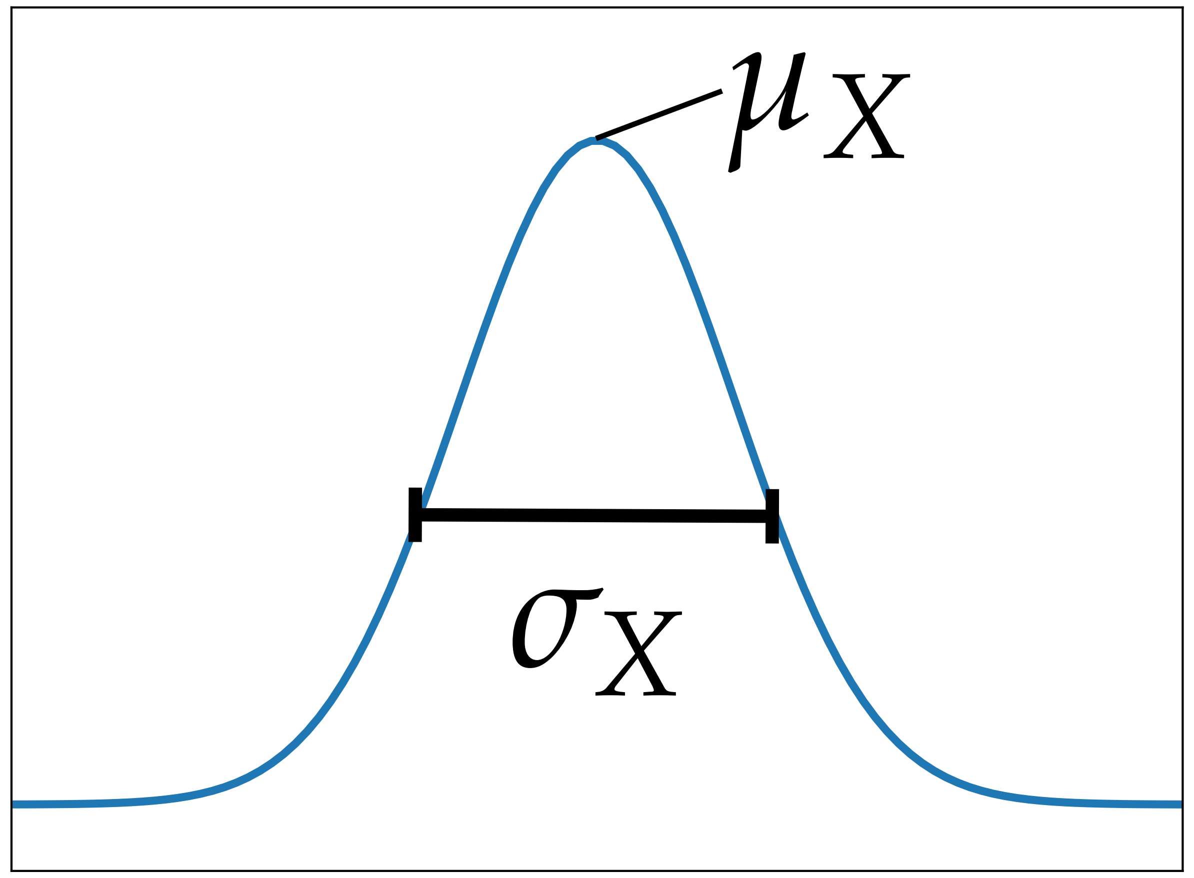
Analysing data in higher dimensions (2D)

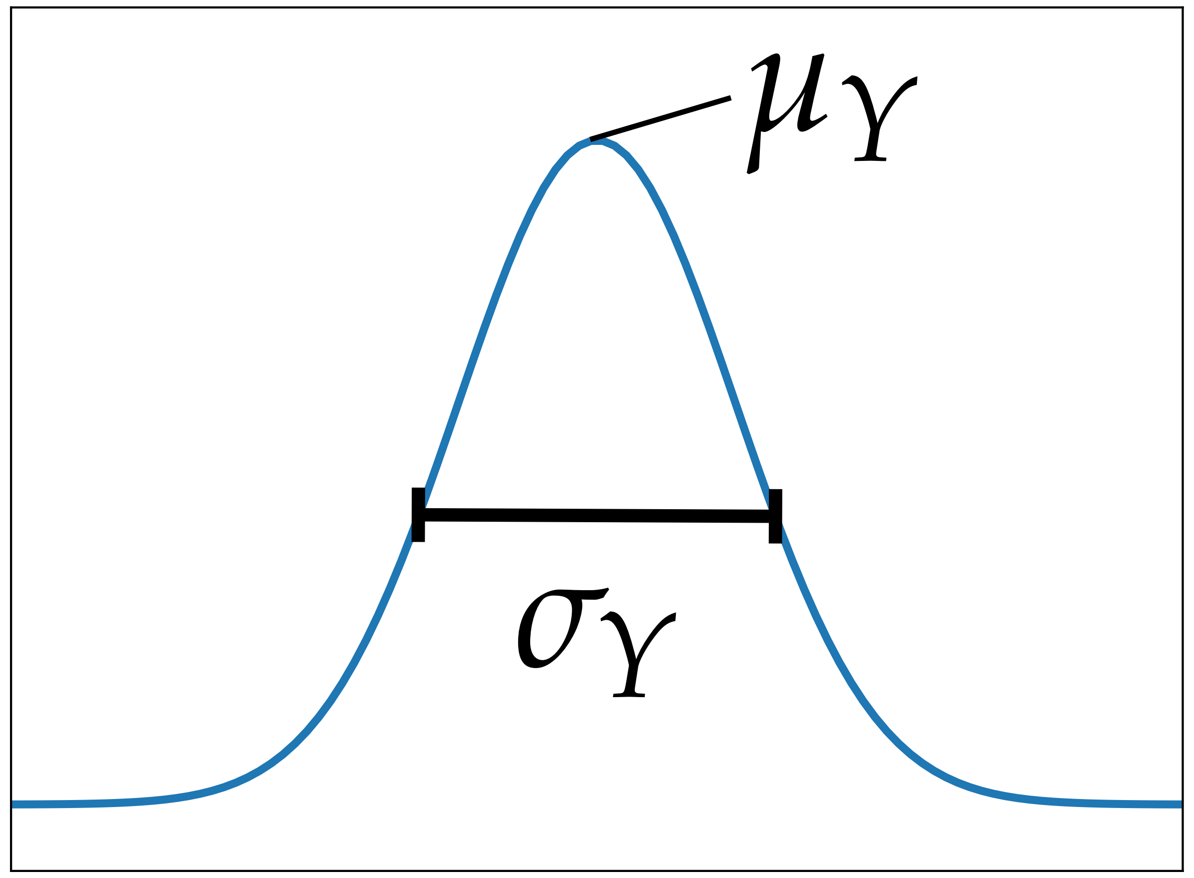
$$X_1, X_2, \ldots \qquad\qquad Y_1, Y_2, \ldots$$

$$\bar{X}_k = \frac{X_1 + \ldots + X_k}{k} \qquad\qquad \bar{Y}_k = \frac{Y_1 + \ldots + Y_k}{k}$$

$$\bar{X}_k \to \mu_X \quad \text{(averages converge to mean)} \quad \bar{Y}_k \to \mu_Y$$

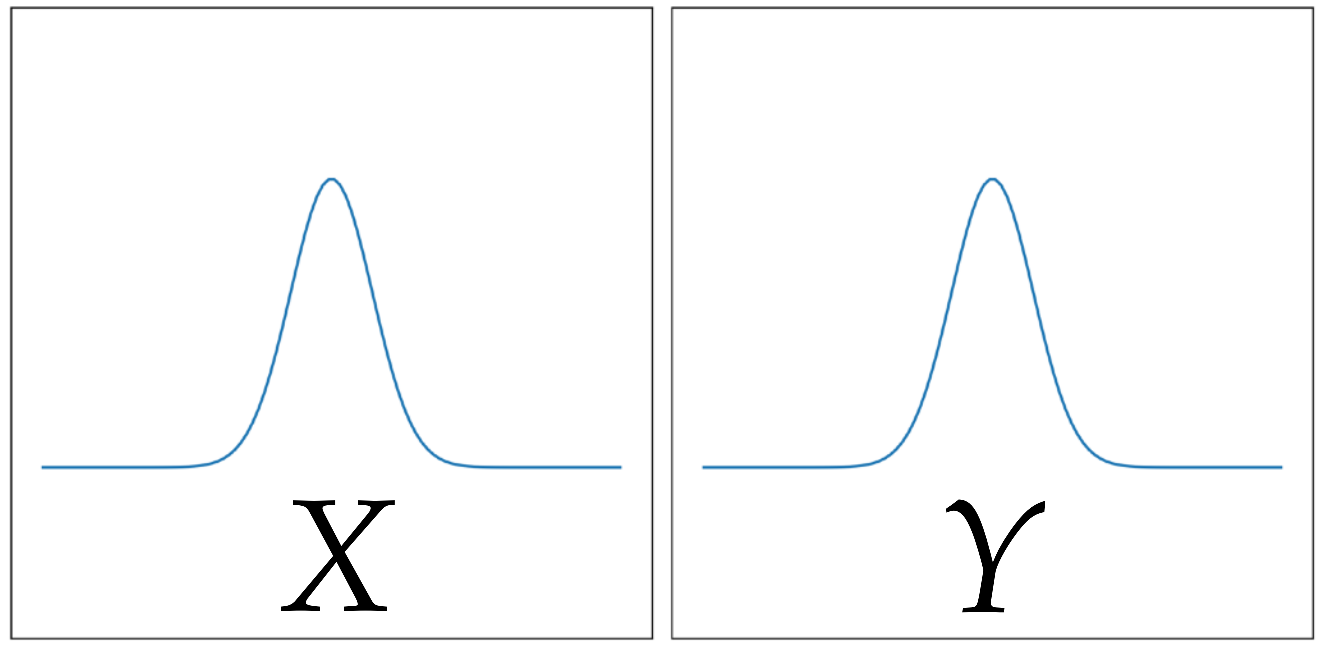
$$\overline{(X_k - \bar{X}_k)^2} \to \sigma_X^2 \quad \text{(squared deviations converge to variance)} \quad \overline{(Y_k - \bar{Y}_k)^2} \to \sigma_Y^2$$



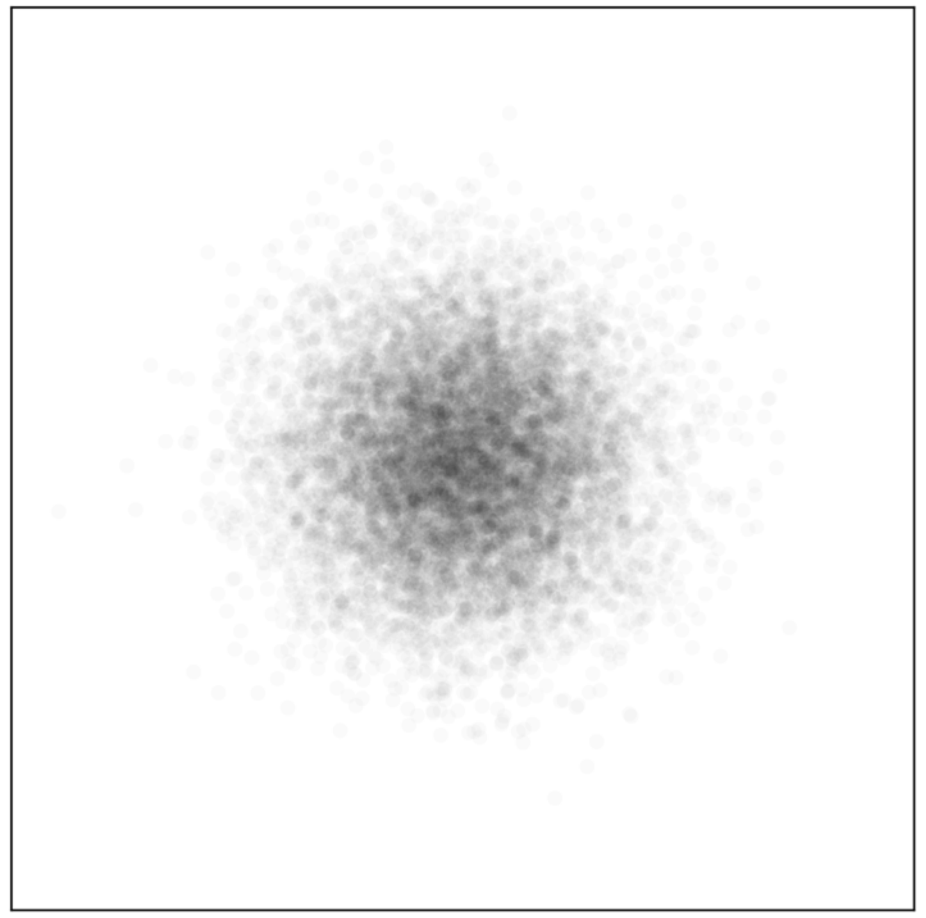


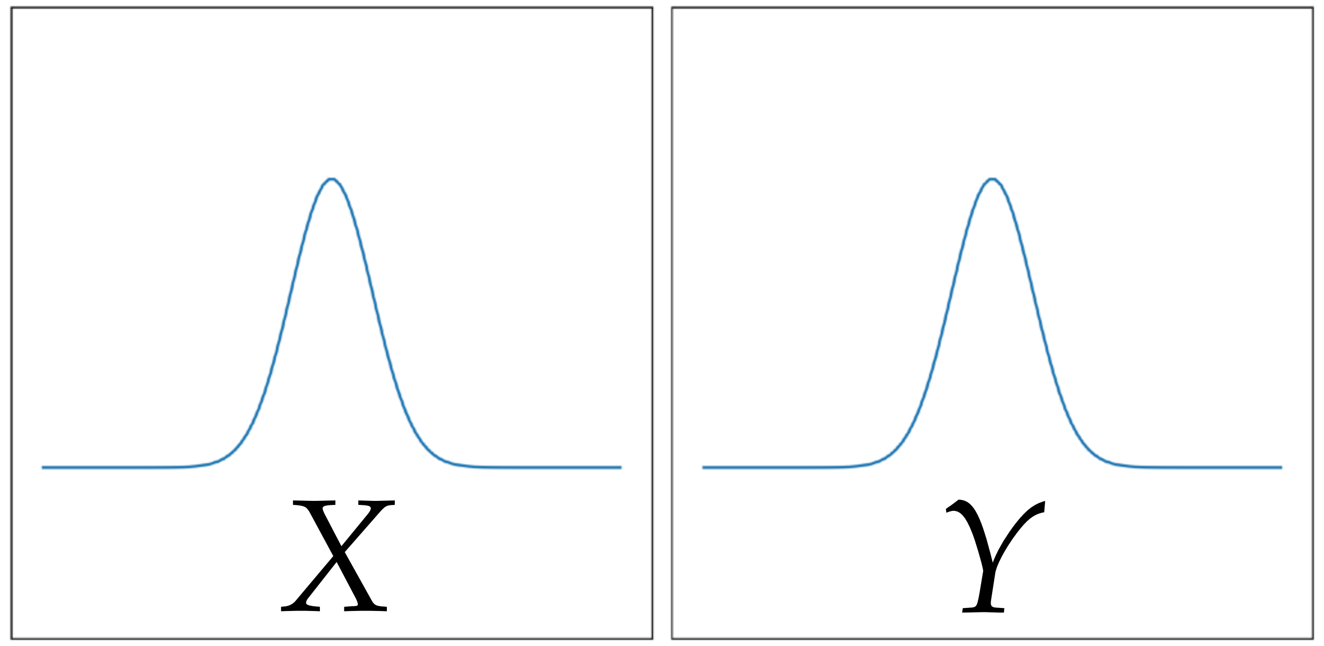
Seeing the whole picture.

Seeing the whole picture.

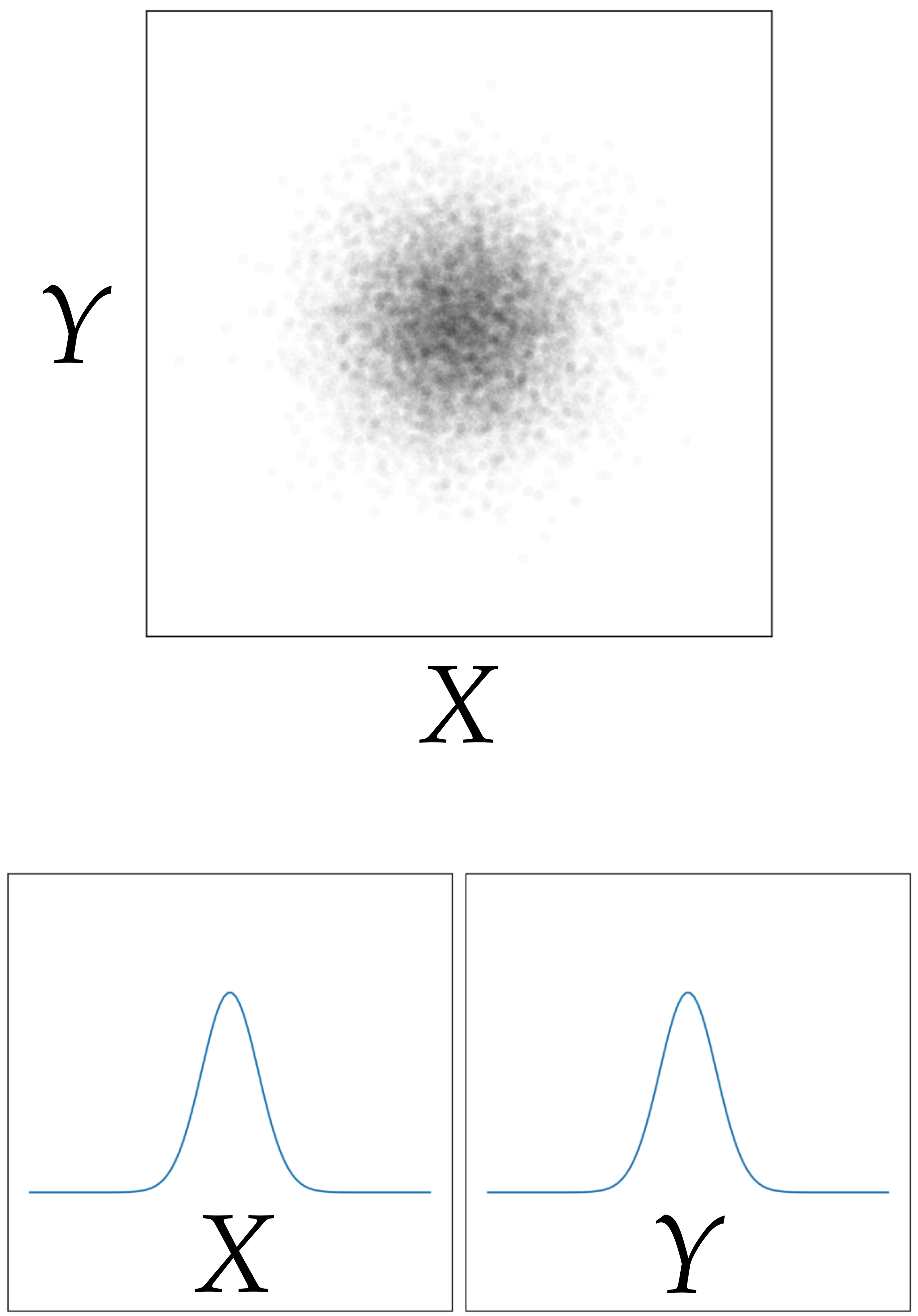


Seeing the whole picture.

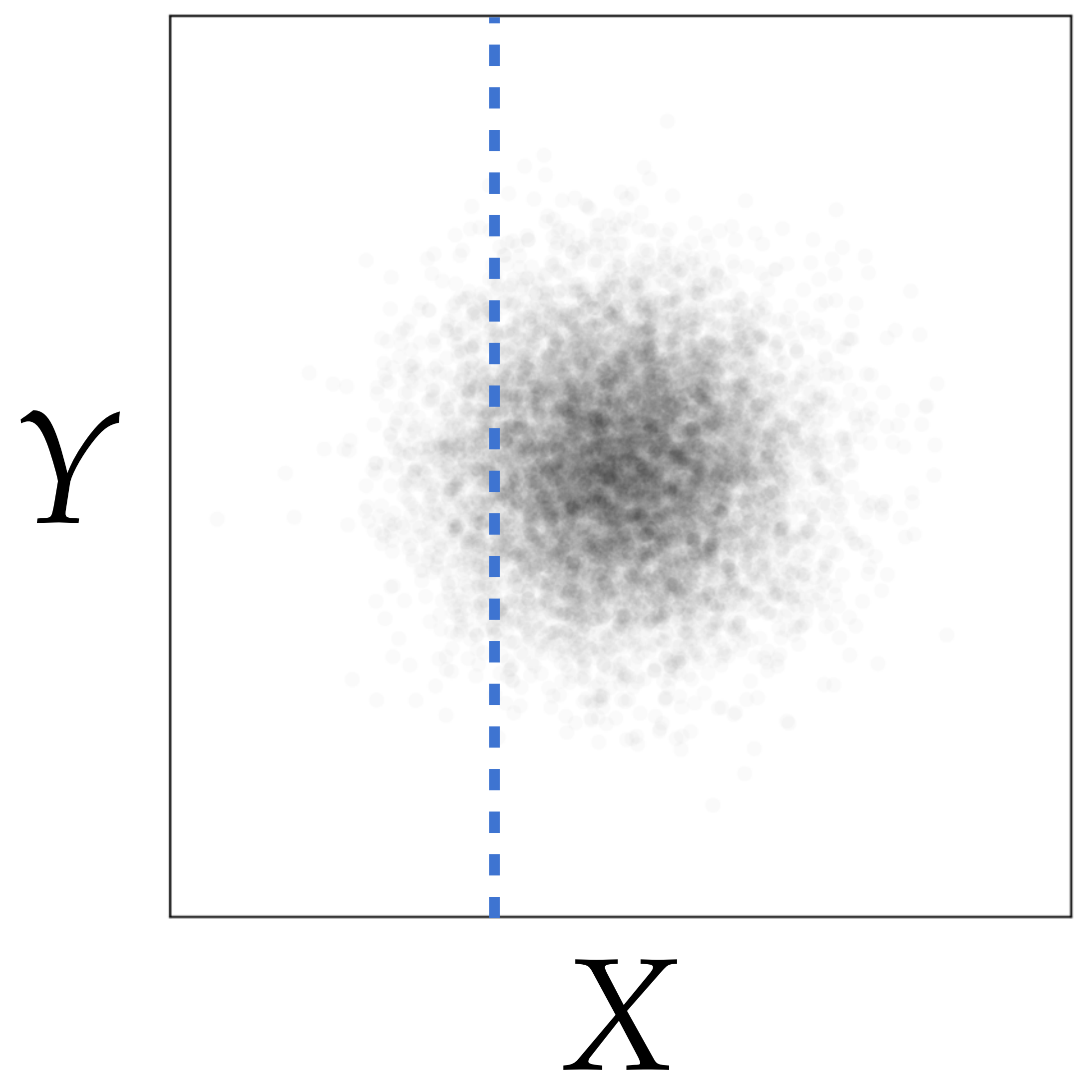


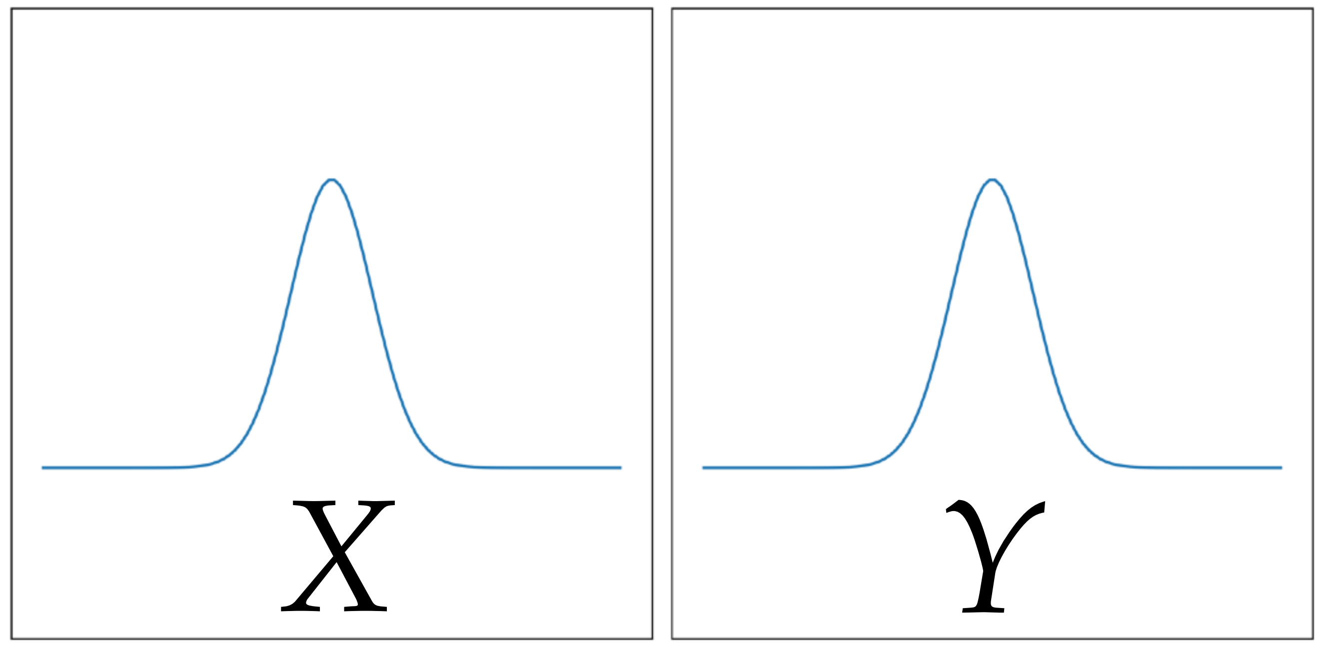


Seeing the whole picture.

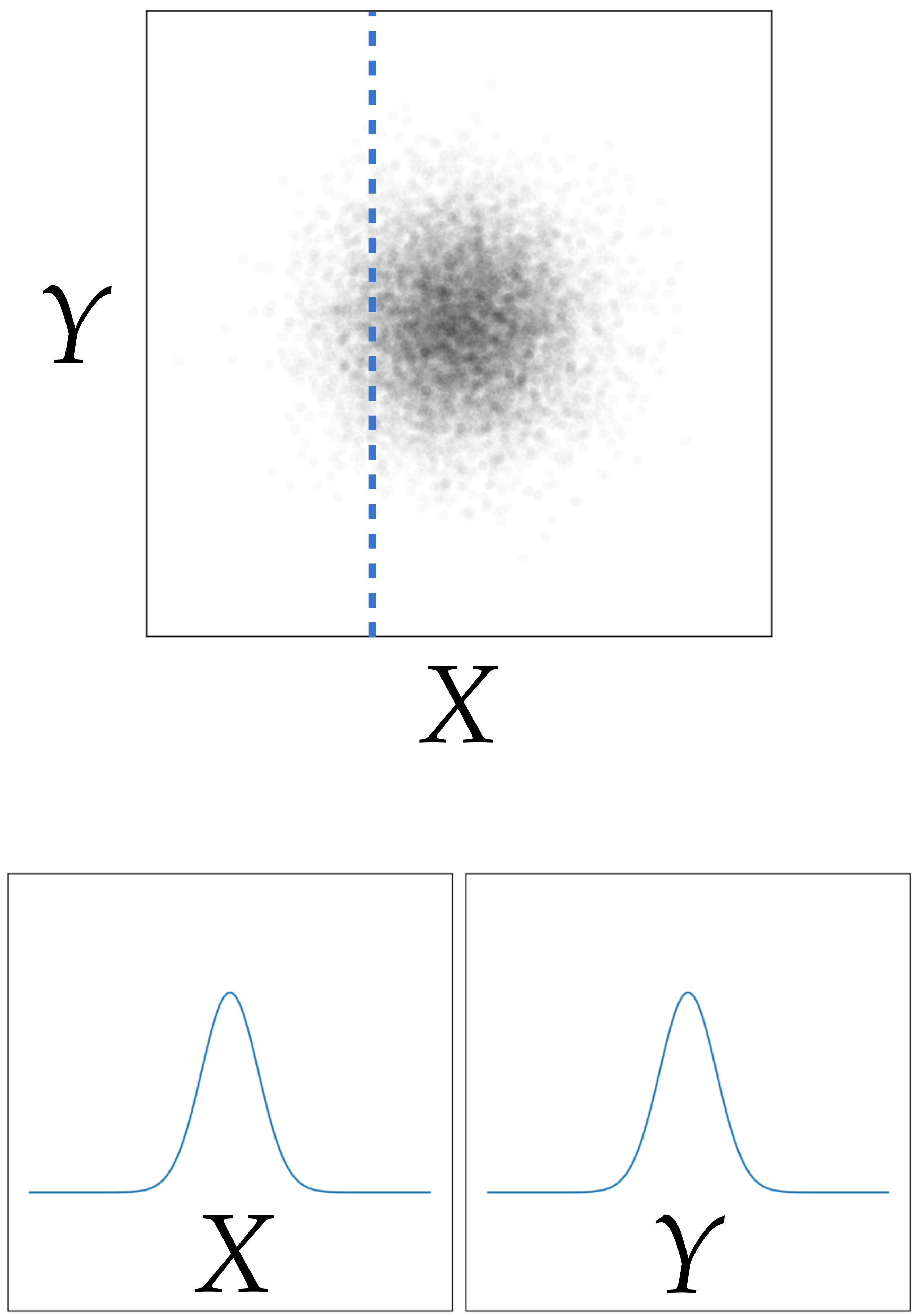


Seeing the whole picture.

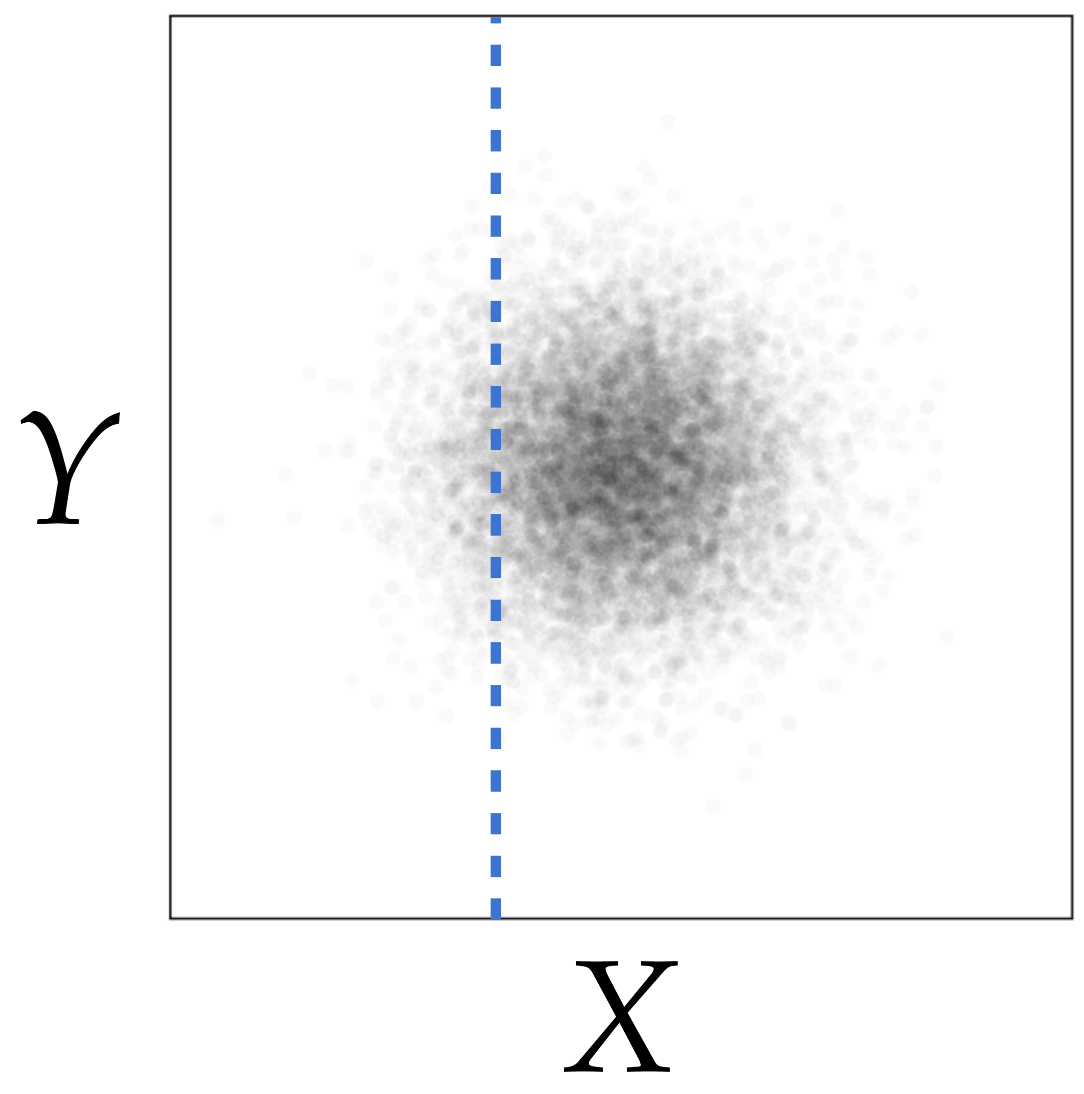


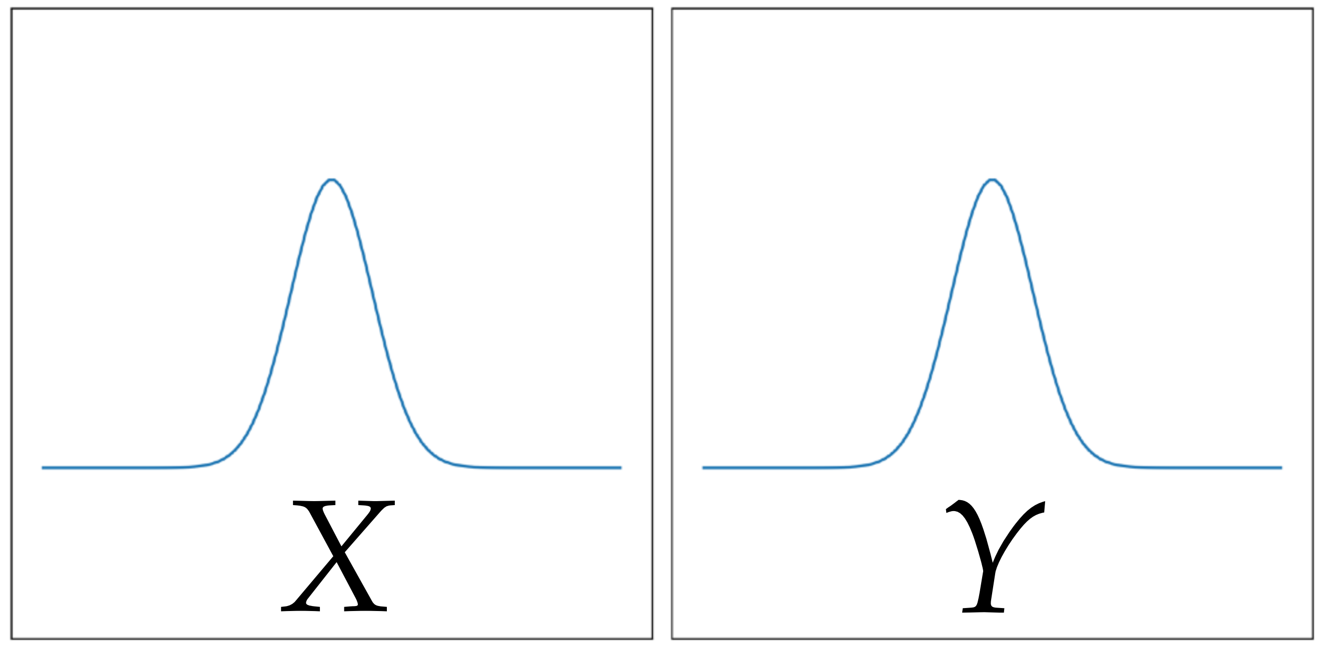


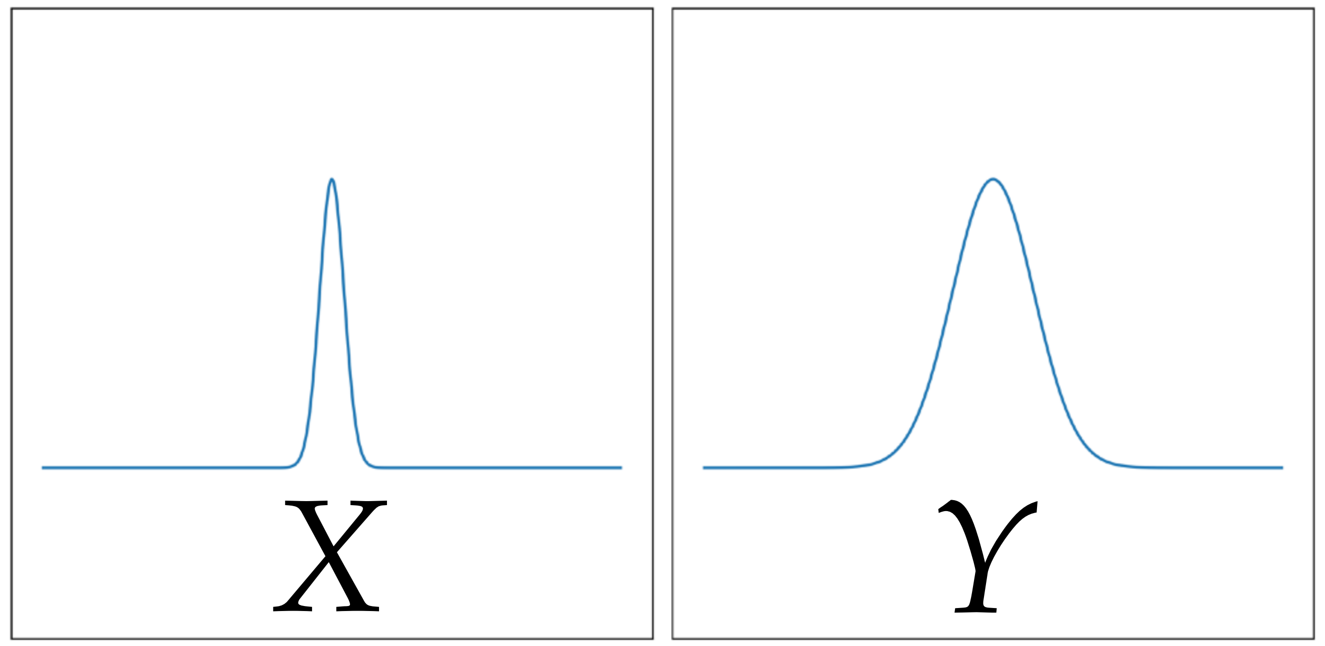
Seeing the whole picture.



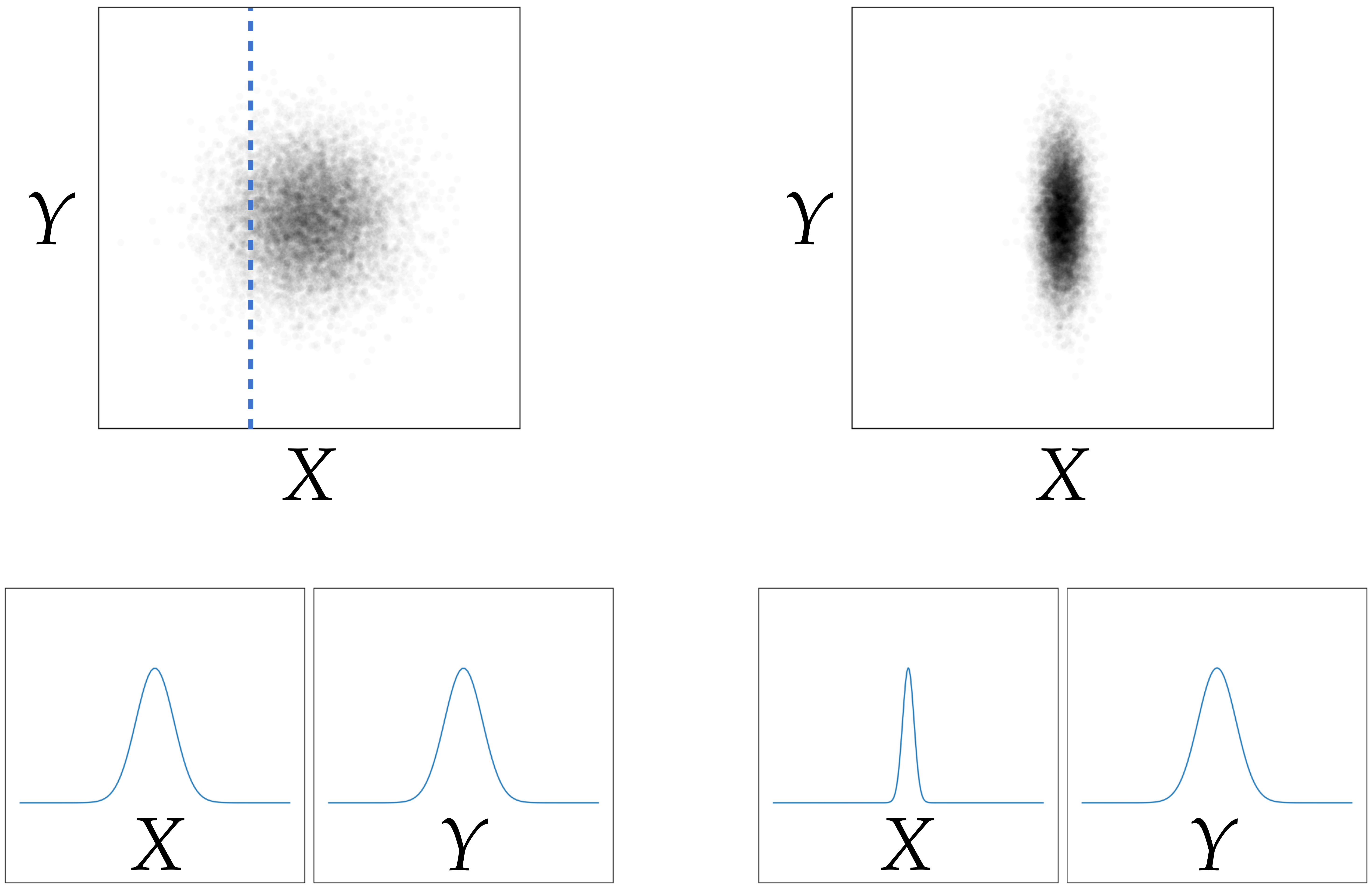
Seeing the whole picture.



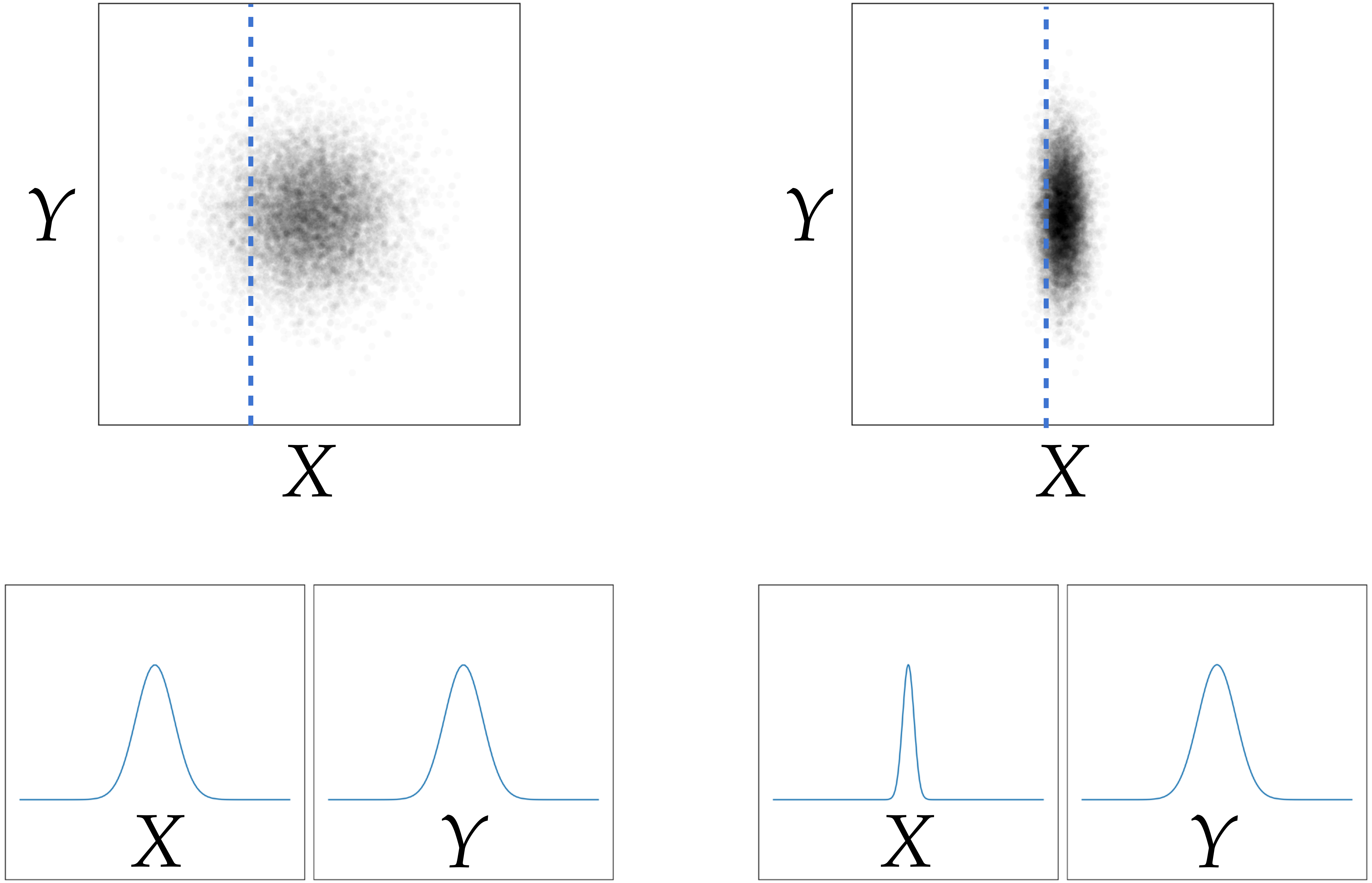




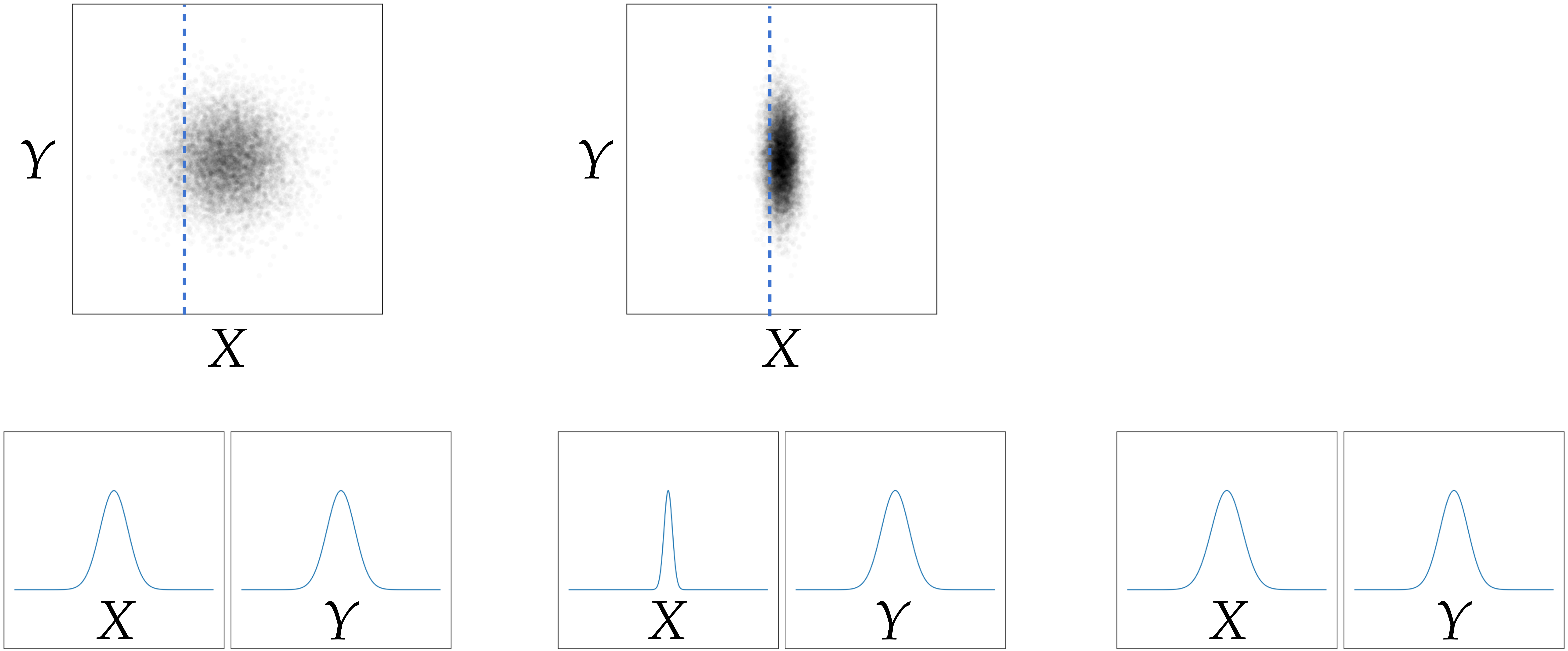
Seeing the whole picture.



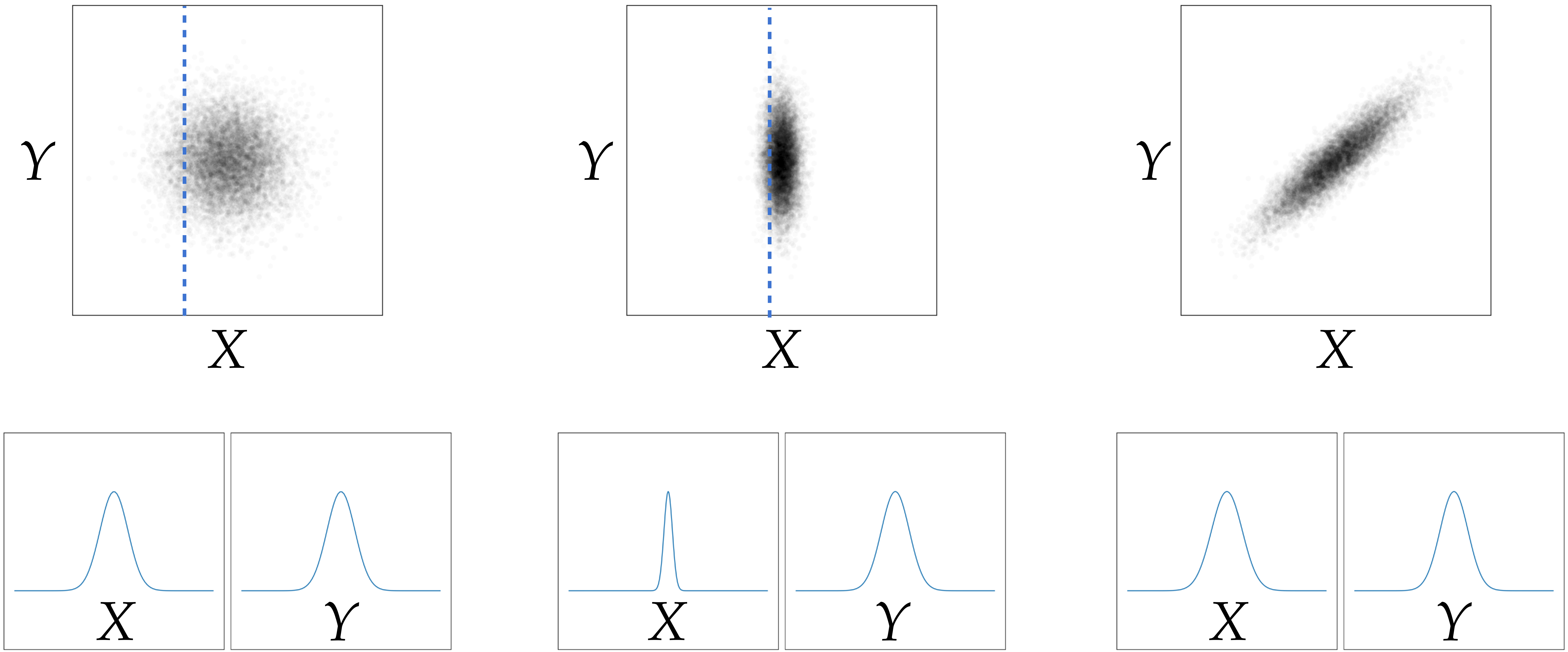
Seeing the whole picture.



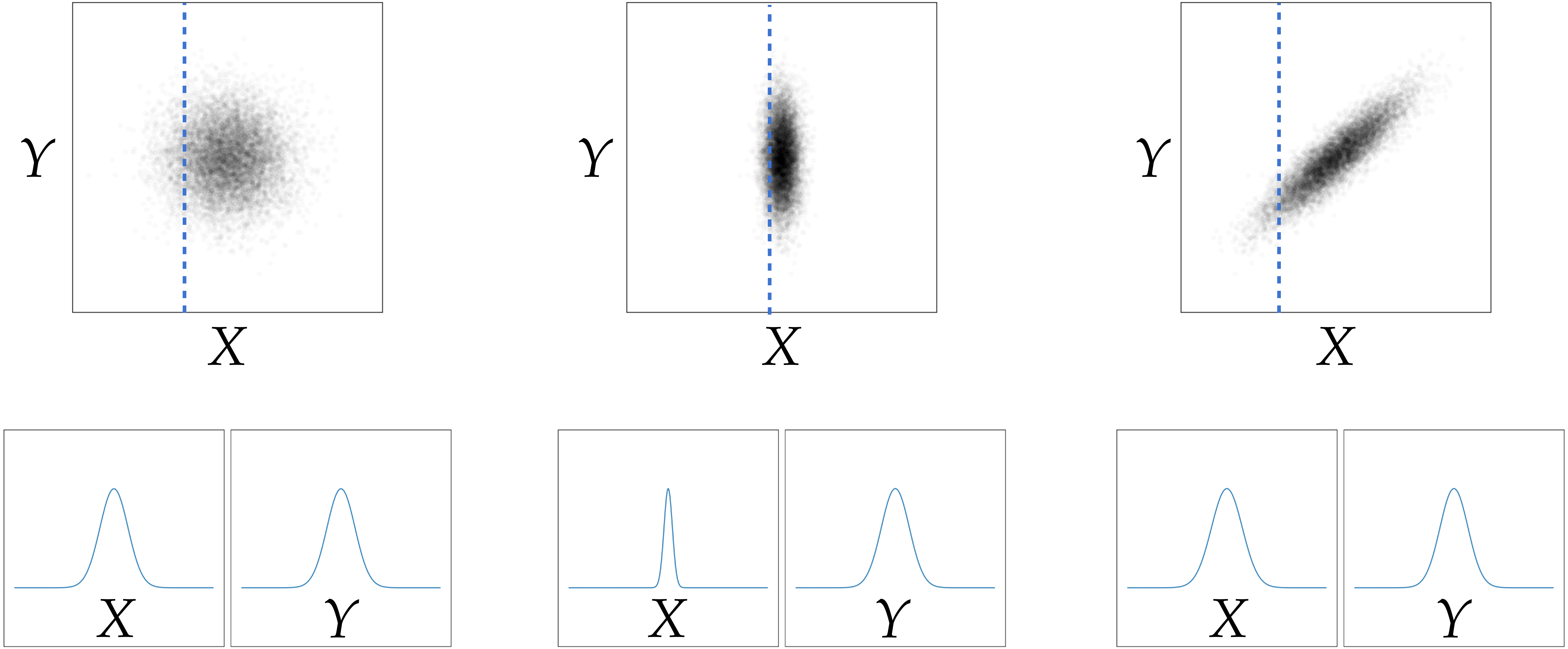
Seeing the whole picture.



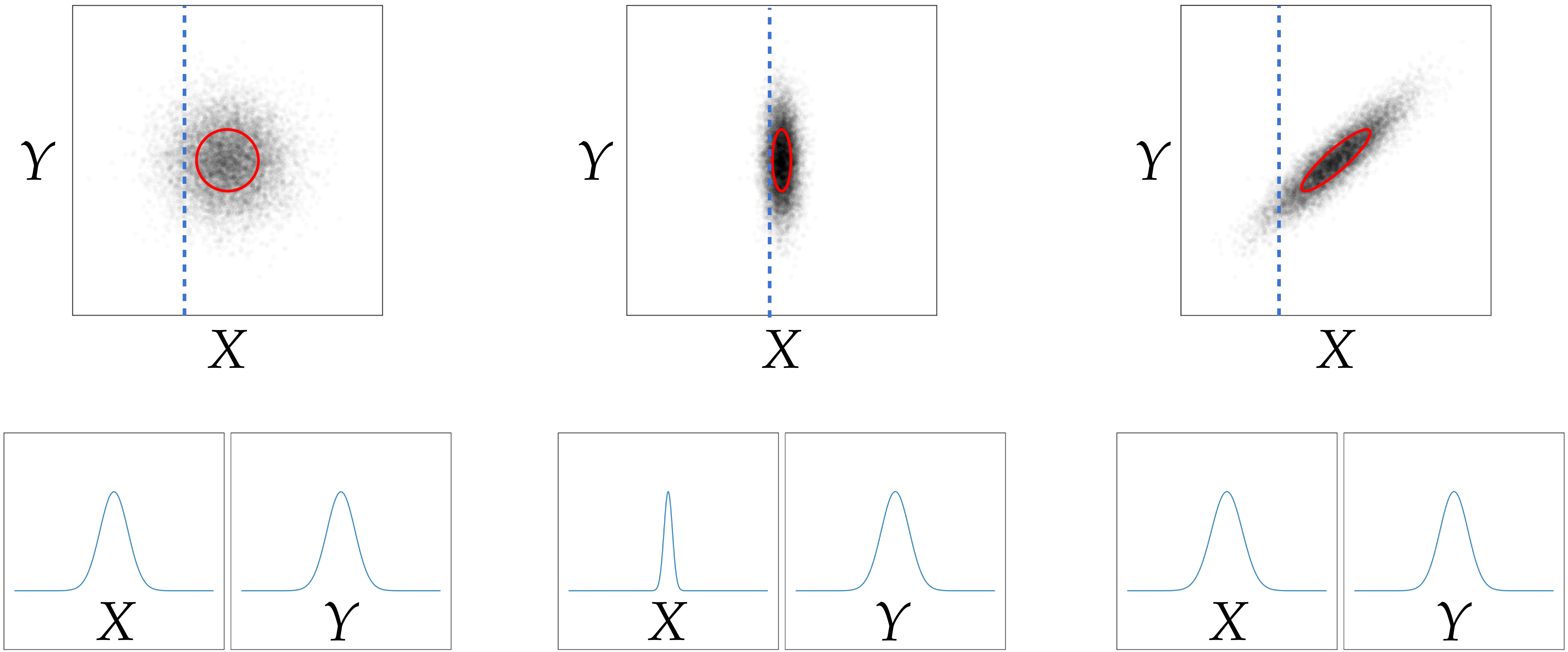
Seeing the whole picture.



Seeing the whole picture.



Seeing the whole picture.



Correlations in multidimensional data

1. Age
2. Hardness
3. Smell intensity
4. Taste intensity
5. Amount of mold
6. Average color (RGB)

MidJourney: *Board with cheese selection, white background.*

Correlations in multidimensional data

1. Age
2. Hardness
3. Smell intensity
4. Taste intensity
5. Amount of mold
6. Average color (RGB)

Intuitively, collecting data will reveal that some fields have an approximately linear relationship (they are **correlated**), e.g.:

- **Very old** cheeses have a **strong taste** and are **smelly**.
- Blue cheeses are generally blue because of mold.

Other axes will show **no correlation** in plots.

MidJourney: *Board with cheese selection, white background.*

Correlations in multidimensional data

1. Age
2. Hardness
3. Smell intensity
4. Taste intensity
5. Amount of mold
6. Average color (RGB)
7. Average number of sun spots during maturation

Intuitively, collecting data will reveal that some fields have an approximately linear relationship (they are **correlated**), e.g.:

- **Very old** cheeses have a **strong taste** and are **smelly**.
- Blue cheeses are generally blue because of mold.

Other axes will show **no correlation** in plots.

MidJourney: *Board with cheese selection, white background.*

Correlations in multidimensional data

1. Age
2. Hardness
3. Smell intensity
4. Taste intensity
5. Amount of mold
6. Average color (RGB)
7. Average number of sun spots during maturation
8. Temperature on the back side of the moon

Intuitively, collecting data will reveal that some fields have an approximately linear relationship (they are **correlated**), e.g.:

- **Very old** cheeses have a **strong taste** and are **smelly**.
- Blue cheeses are generally blue because of mold.

Other axes will show **no correlation** in plots.

MidJourney: *Board with cheese selection, white background.*

Covariance: in 2 dimensions

Definition: $\mathrm{Cov}(X, Y) = E\left[(X - E[X])(Y - E[Y])\right]$

Covariance: in 2 dimensions

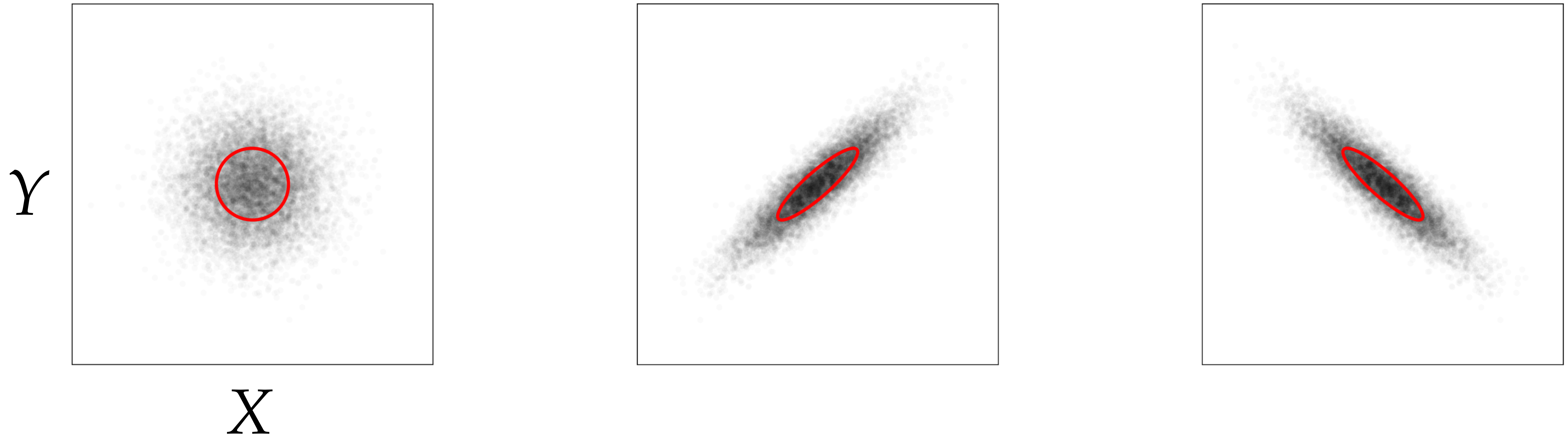
Definition: $$\mathrm{Cov}(X,Y) = E\left[(X - E[X])(Y - E[Y])\right]$$

Estimator: $$\overline{(X_k - \bar{X})(Y_k - \bar{Y})} \to \mathrm{Cov}(X,Y) \quad (k \to \infty)$$

Covariance: in 2 dimensions

Definition: $$\mathrm{Cov}(X,Y) = E\left[(X - E[X])(Y - E[Y])\right]$$

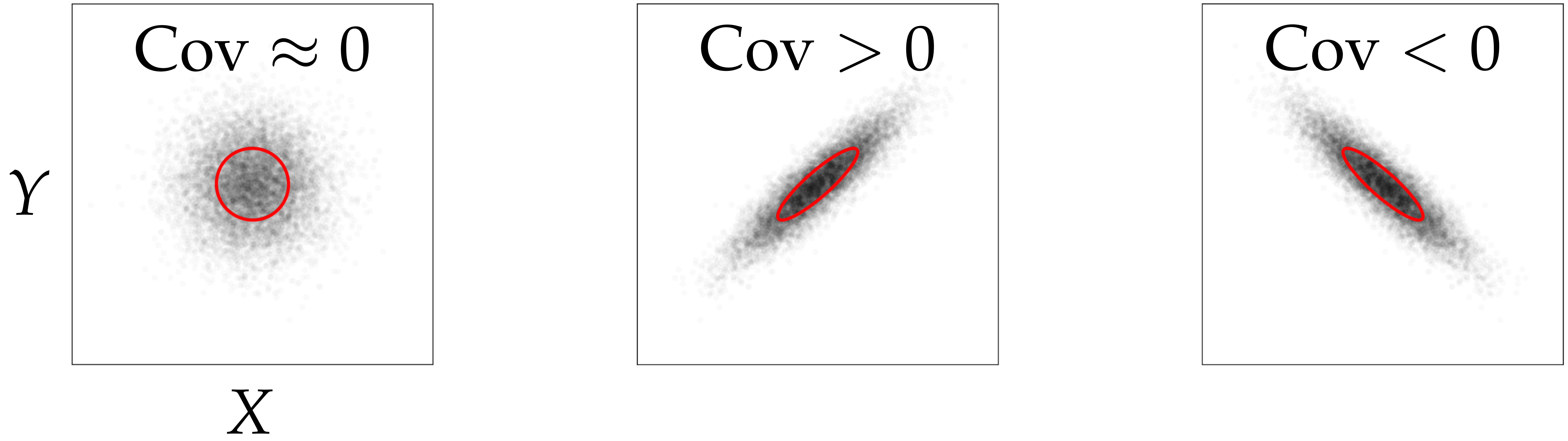
Estimator: $$\overline{(X_k - \bar{X})(Y_k - \bar{Y})} \to \mathrm{Cov}(X,Y) \quad (k \to \infty)$$



Covariance: in 2 dimensions

Definition: $$\mathrm{Cov}(X,Y) = E\left[(X - E[X])(Y - E[Y])\right]$$

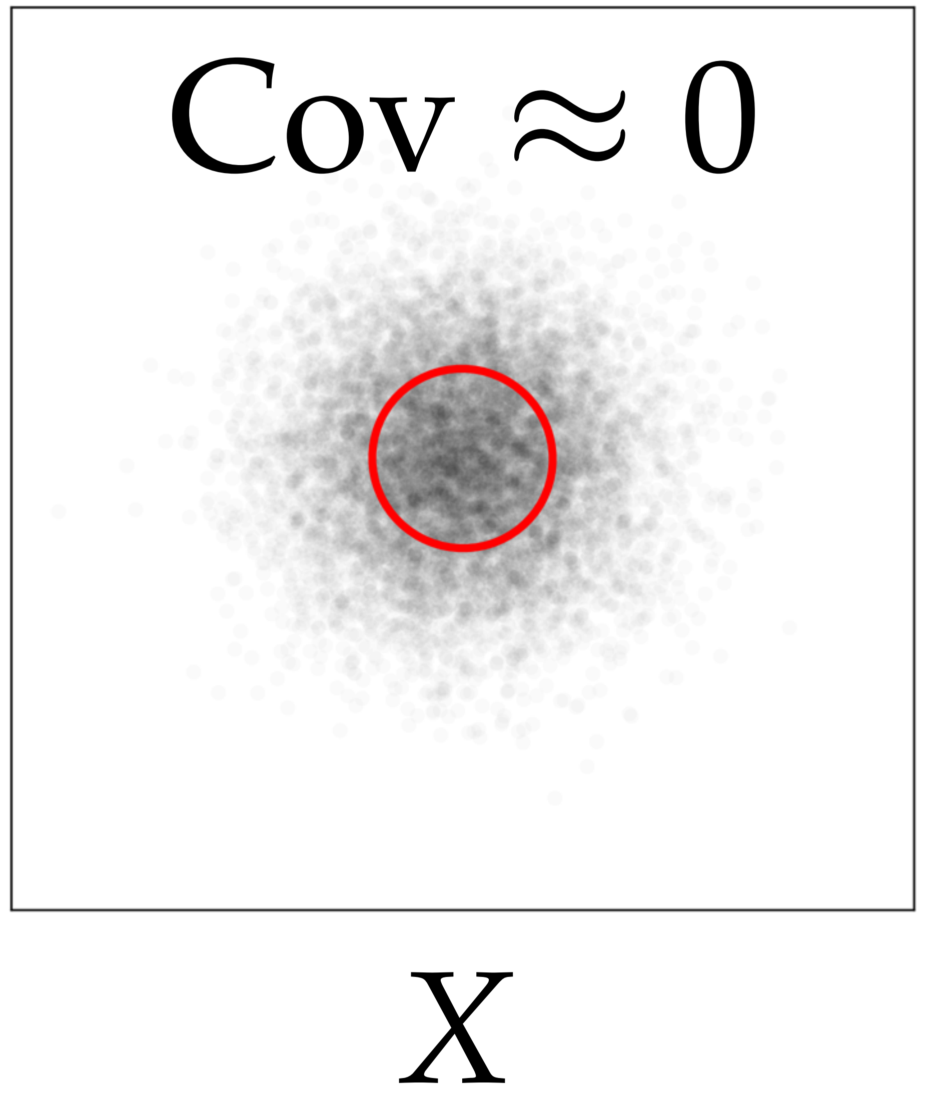
Estimator: $$\overline{(X_k - \bar{X})(Y_k - \bar{Y})} \to \mathrm{Cov}(X,Y) \quad (k \to \infty)$$

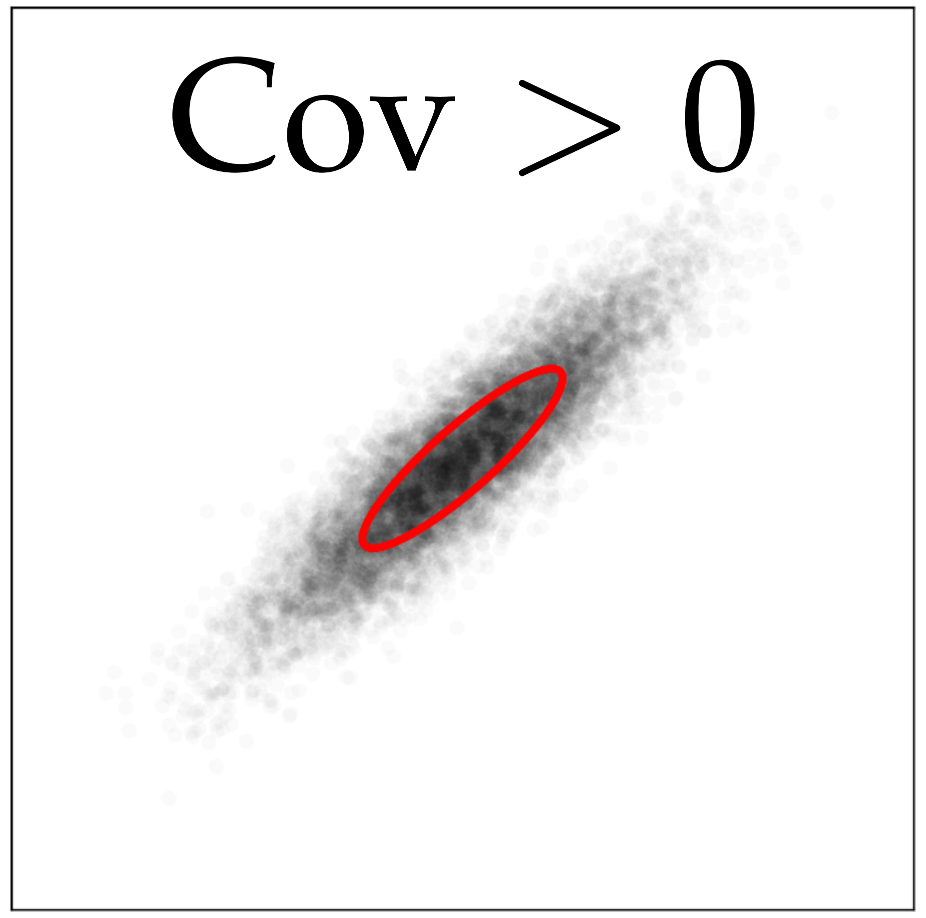


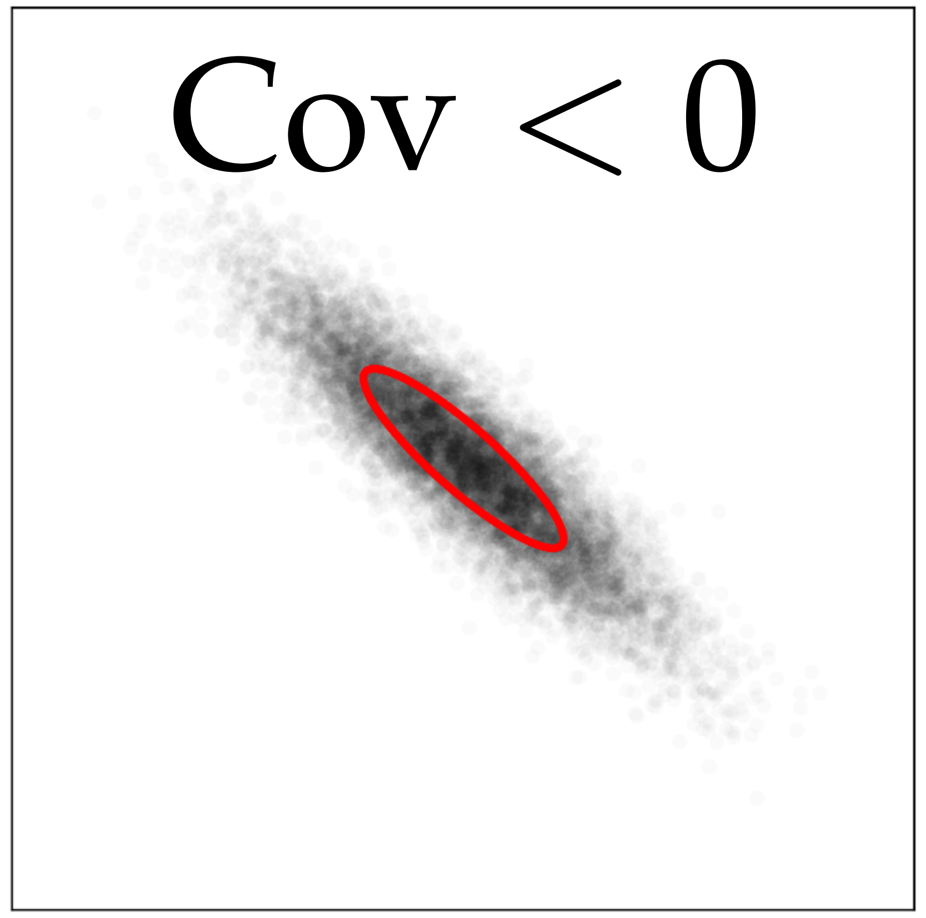
Covariance: in 2 dimensions

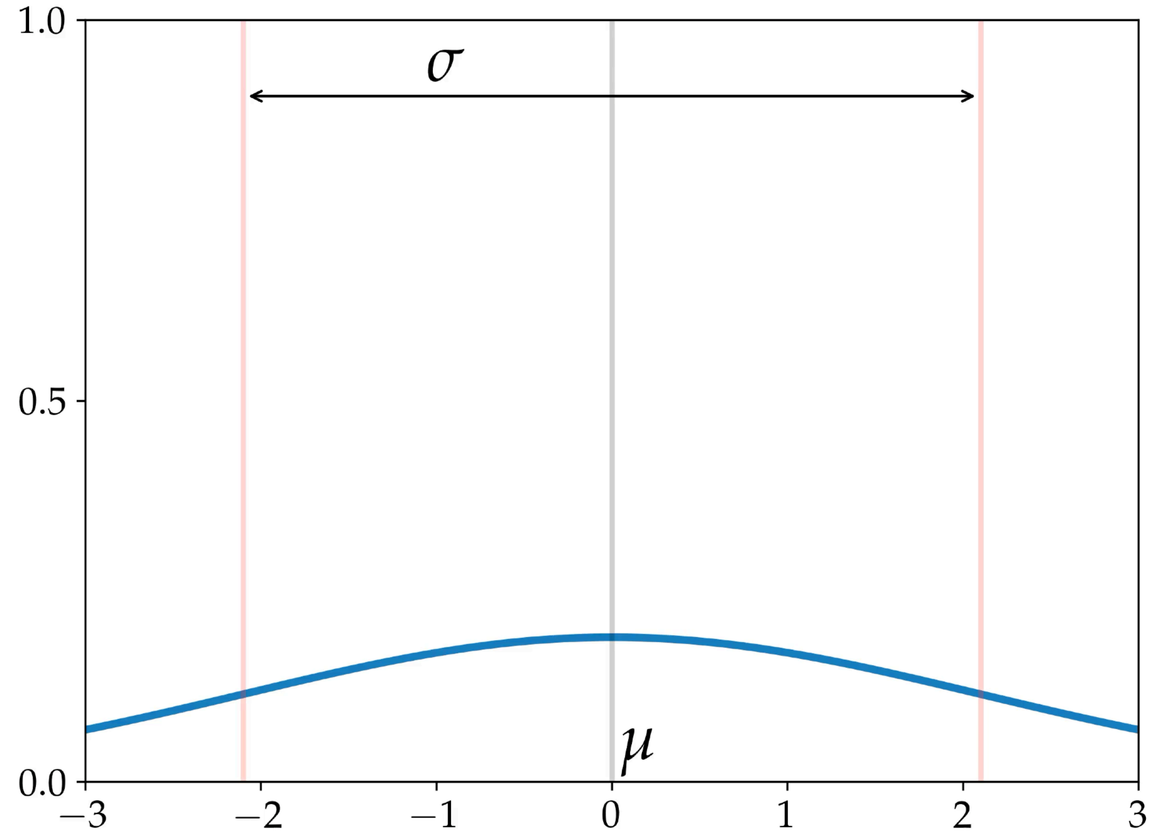
Definition: $\mathrm{Cov}(X,Y) = E\left[(X - E[X])(Y - E[Y])\right]$

Estimator: $\overline{(X_k - \bar{X})(Y_k - \bar{Y})} \to \mathrm{Cov}(X,Y) \quad (k \to \infty)$









Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & & \\ & & \\ & & \end{bmatrix}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} \\ X_1 - \bar{X} \\ \vdots \\ X_n - \bar{X} \end{bmatrix}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & \\ X_1 - \bar{X} & Y_1 - \bar{Y} & \\ \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & \end{bmatrix}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & Z_0 - \bar{Z} & \cdots \\ X_1 - \bar{X} & Y_1 - \bar{Y} & Z_1 - \bar{Z} & \cdots \\ \vdots & \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & Z_n - \bar{Z} & \cdots \end{bmatrix}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & Z_0 - \bar{Z} & \cdots \\ X_1 - \bar{X} & Y_1 - \bar{Y} & Z_1 - \bar{Z} & \cdots \\ \vdots & \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & Z_n - \bar{Z} & \cdots \end{bmatrix}$$

$$\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & Z_0 - \bar{Z} & \cdots \\ X_1 - \bar{X} & Y_1 - \bar{Y} & Z_1 - \bar{Z} & \cdots \\ \vdots & \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & Z_n - \bar{Z} & \cdots \end{bmatrix}$$

n observations

$$\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & Z_0 - \bar{Z} & \cdots \\ X_1 - \bar{X} & Y_1 - \bar{Y} & Z_1 - \bar{Z} & \cdots \\ \vdots & \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & Z_n - \bar{Z} & \cdots \end{bmatrix}$$

n observations

m dimensions

$$\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

Covariance: in *m* dimensions

$$\mathbf{X} = \begin{bmatrix} X_0 - \bar{X} & Y_0 - \bar{Y} & Z_0 - \bar{Z} & \cdots \\ X_1 - \bar{X} & Y_1 - \bar{Y} & Z_1 - \bar{Z} & \cdots \\ \vdots & \vdots & \vdots & \\ X_n - \bar{X} & Y_n - \bar{Y} & Z_n - \bar{Z} & \cdots \end{bmatrix}$$

n observations

m dimensions

$$\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

Sometimes *N-1* is used here instead. It depends on the setup (beyond the scope of CS328).

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$$

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean

Covariance matrix

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$

Covariance matrix $\in \mathbb{R}^{n \times n}$

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$

Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$$

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$

Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

normalization constant

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$

Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

normalization constant

exponent

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$ Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

normalization constant exponent

Compare to 1D case:

$$f_x(x) = (2\pi)^{-\frac{1}{2}} \sigma^{-1} \exp\left(-\frac{1}{2}(x-\mu)\sigma^{-2}(x-\mu)\right)$$

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$ Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

normalization constant exponent

Compare to 1D case:

$$f_x(x) = (2\pi)^{-\frac{1}{2}} \sigma^{-1} \exp\left(-\frac{1}{2}(x-\mu)\sigma^{-2}(x-\mu)\right)$$

Multivariate Normal Distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mean $\in \mathbb{R}^n$ Covariance matrix $\in \mathbb{R}^{n \times n}$

Density function:

$$f_{\mathbf{x}}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

normalization constant exponent

Compare to 1D case:

$$f_x(x) = (2\pi)^{-\frac{1}{2}} \sigma^{-1} \exp\left(-\frac{1}{2}(x-\mu)\sigma^{-2}(x-\mu)\right)$$

Statistical Analysis of Multivariate Data

Principal component analysis (PCA): using eigenvectors and eigenvalues to describe data.

- **Step 1**: compute the mean of all data points. Subtract this value from all data points.

Statistical Analysis of Multivariate Data

Principal component analysis (PCA): using eigenvectors and eigenvalues to describe data.

- **Step 1**: compute the mean of all data points. Subtract this value from all data points.

- *Version 1* (via Eigenanalysis):
  - **Step 2**: estimate the covariance matrix $\mathbf{\Sigma} = \mathbf{X}^T\mathbf{X} / n$
  - **Step 3:** compute $\mathbf{V}, \mathbf{\Lambda} = \text{eig}(\mathbf{\Sigma})$
  - **Profit**: principal directions given by $\mathbf{V}$, radii given by $\sqrt{\mathbf{\Lambda}}$

Statistical Analysis of Multivariate Data

Principal component analysis (PCA): using eigenvectors and eigenvalues to describe data.

- **Step 1**: compute the mean of all data points. Subtract this value from all data points.

- *Version 1* (via Eigenanalysis):
  - **Step 2**: estimate the covariance matrix $\mathbf{\Sigma} = \mathbf{X}^T\mathbf{X} \,/\, n$
  - **Step 3:** compute $\mathbf{V}, \mathbf{\Lambda} = \mathrm{eig}(\mathbf{\Sigma})$
  - **Profit**: principal directions given by $\mathbf{V}$, radii given by $\sqrt{\mathbf{\Lambda}}$

Statistical Analysis of Multivariate Data

Principal component analysis (PCA): using eigenvectors and eigenvalues to describe data.

- **Step 1**: compute the mean of all data points. Subtract this value from all data points.

- *Version 1* (via Eigenanalysis):
  - **Step 2**: estimate the covariance matrix $\mathbf{\Sigma} = \mathbf{X}^T\mathbf{X} / n$
  - **Step 3:** compute $\mathbf{V}, \mathbf{\Lambda} = \mathrm{eig}(\mathbf{\Sigma})$
  - **Profit**: principal directions given by **V**, radii given by $\sqrt{\mathbf{\Lambda}}$

- *Version 2* **(via SVD):**
  - **Step 2**: compute $\mathbf{U}, \mathbf{S}, \mathbf{V} = \mathrm{svd}(\mathbf{X} / \sqrt{n})$
  - **Profit**: principal directions given by **V**, radii given by **S**

Creating Normal Variates

Creating general normal variates (a.k.a. "simulating" or "sampling" them)

1-D case: $$X \sim \mathcal{N}(0,1) \quad \Rightarrow \quad \sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$$

Creating Normal Variates

Creating general normal variates (a.k.a. "simulating" or "sampling" them)

1-D case: $X \sim \mathcal{N}(0,1) \quad \Rightarrow \quad \sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$

NumPy recipe:

```
sigma * np.random.randn() + mu
```

Creating Normal Variates

Creating general normal variates (a.k.a. "simulating" or "sampling" them)

1-D case: $X \sim \mathcal{N}(0,1) \quad \Rightarrow \quad \sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$

NumPy recipe:

```
sigma * np.random.randn() + mu
```

N-D case: $\mathbf{X} \sim \mathcal{N}(0,\mathbf{I}) \quad \Rightarrow \quad \mathbf{A}\mathbf{X} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{A}\mathbf{A}^T)$

Creating Normal Variates

Creating general normal variates (a.k.a. "simulating" or "sampling" them)

1-D case: $X \sim \mathcal{N}(0,1) \quad \Rightarrow \quad \sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$

NumPy recipe:

```
sigma * np.random.randn() + mu
```

N-D case: $\mathbf{X} \sim \mathcal{N}(0,\mathbf{I}) \quad \Rightarrow \quad \mathbf{A}\mathbf{X} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{A}\mathbf{A}^T)$

$$\Downarrow$$

$$\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{X} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Creating Normal Variates

Creating general normal variates (a.k.a. "simulating" or "sampling" them)

1-D case: $X \sim \mathcal{N}(0,1) \quad \Rightarrow \quad \sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$

NumPy recipe:

```
sigma * np.random.randn() + mu
```

N-D case: $\mathbf{X} \sim \mathcal{N}(0,\mathbf{I}) \quad \Rightarrow \quad \mathbf{A}\mathbf{X} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{A}\mathbf{A}^T)$

$$\Downarrow$$

$$\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{X} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

NumPy recipe:

```
sqrt_sigma @ np.random.randn(n) + mu
```

Demo time

The Outer Product

Also often called a rank-1 matrix.

$$\mathbf{uv}^T = \begin{bmatrix} u_1v_1 & u_1v_2 & \dots & u_1v_n \\ u_2v_1 & u_2v_2 & \dots & u_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_mv_1 & u_mv_2 & \dots & u_mv_n \end{bmatrix}$$

Another view of the SVD

$$\begin{bmatrix} \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \end{bmatrix} \begin{bmatrix} \ddots \boldsymbol{\Sigma} \ddots \end{bmatrix} \begin{bmatrix} \mathbf{V}^T \end{bmatrix}$$

Another view of the SVD

$$\left[\mathbf{A}\right] = \left[\mathbf{U}\right]\left[\boldsymbol{\Sigma}\right]\left[\mathbf{V}^T\right]$$

where

$$\mathbf{V} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}$$

Another view of the SVD

$$\mathbf{A} = \mathbf{U} \, \mathbf{\Sigma} \, \mathbf{V}^T$$

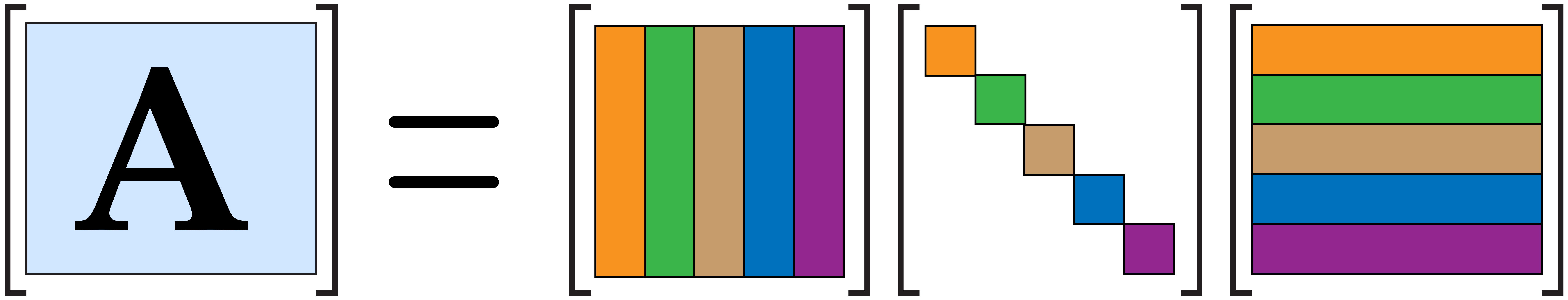
where

$$\mathbf{V} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_n \\ | & & | \end{pmatrix} \quad \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix}$$

Then $\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$

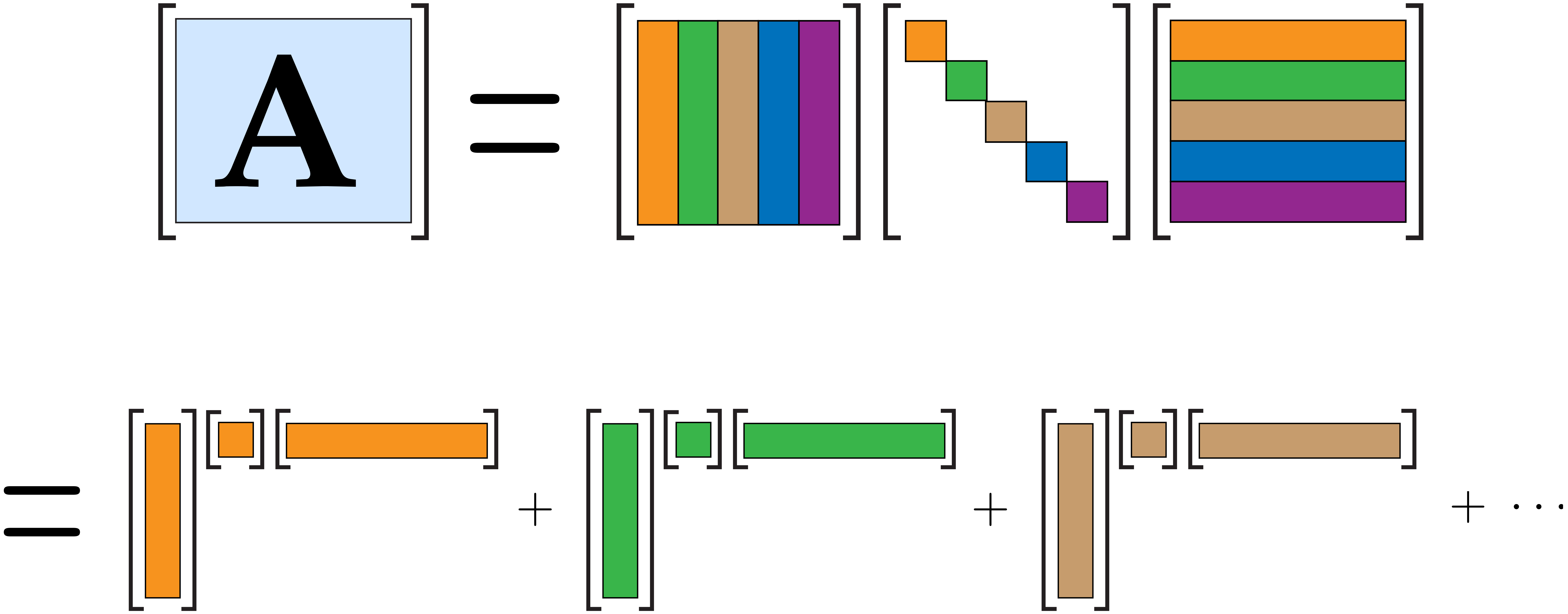
Another view of the SVD

Coloring each combination of left/right singular vector & value reveals outer product structure



Another view of the SVD

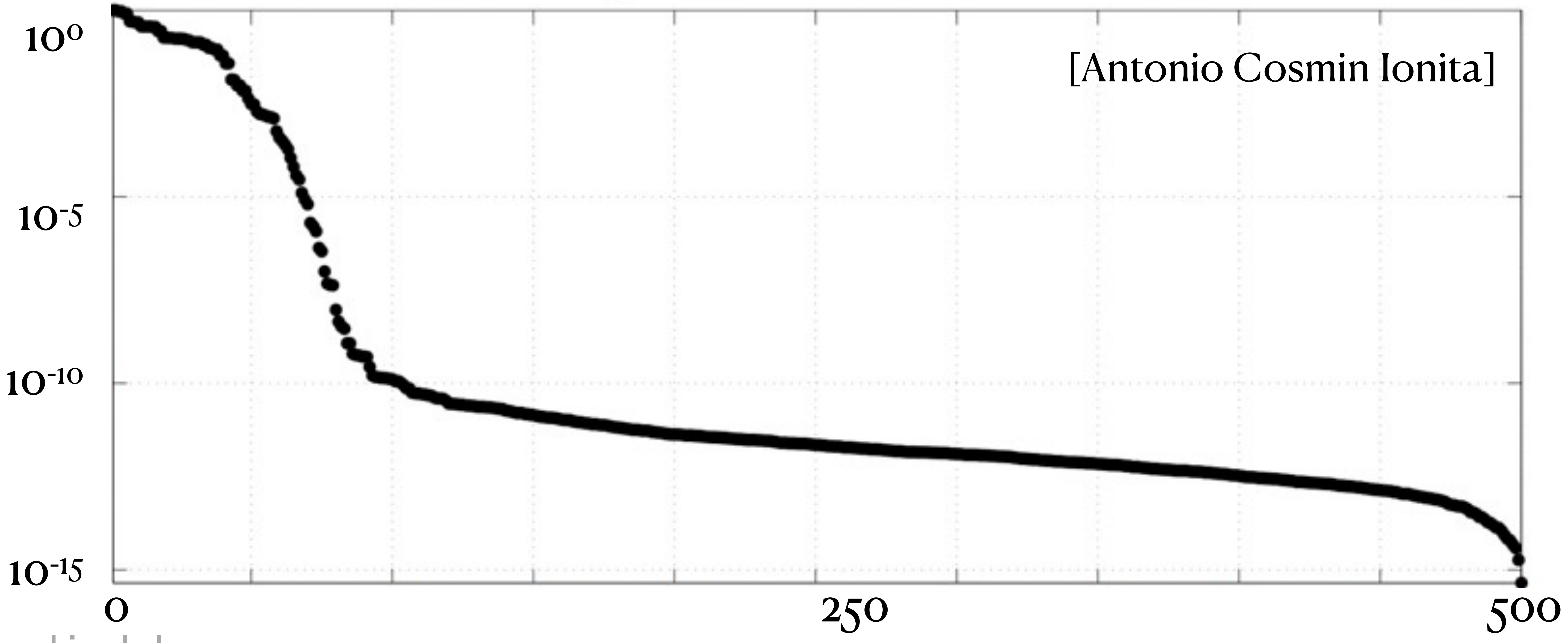
Coloring each combination of left/right singular vector & value reveals outer product structure



Matrix approximation

$$\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

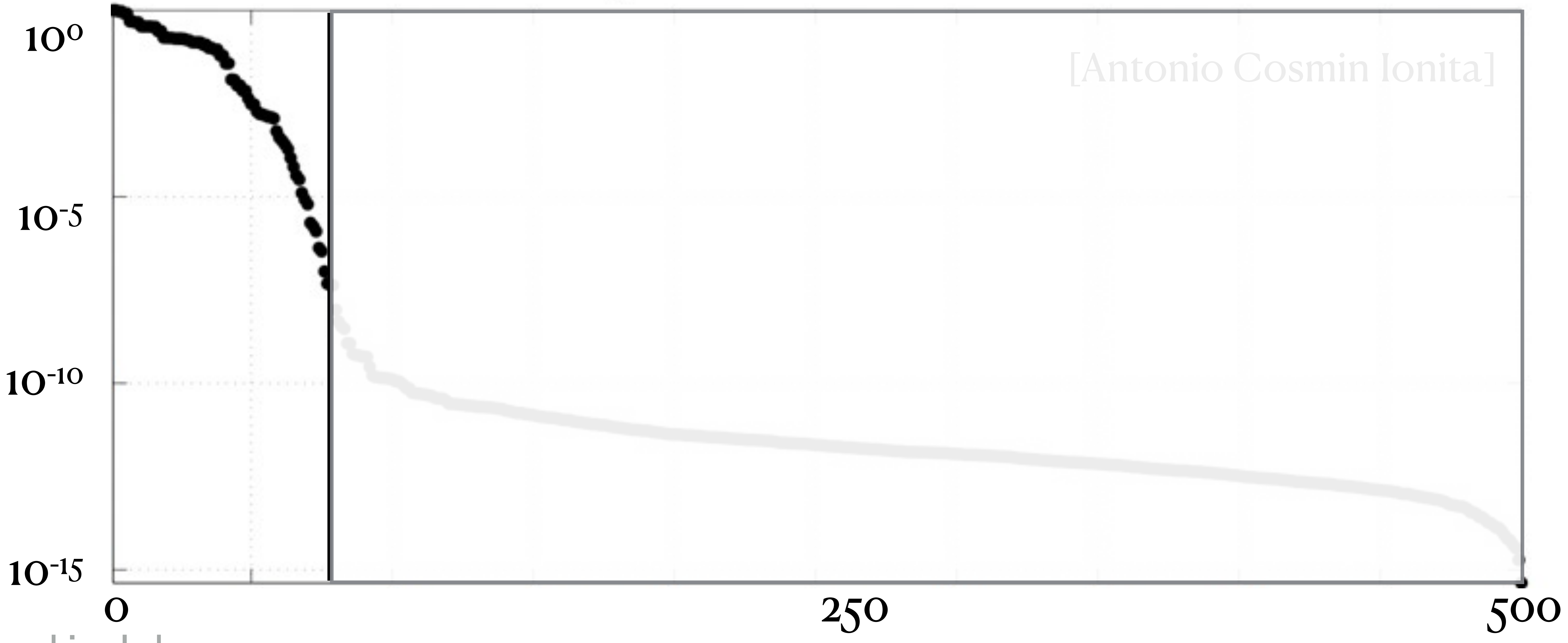
Plot of singular values (in decreasing magnitude)



Matrix approximation

$$\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

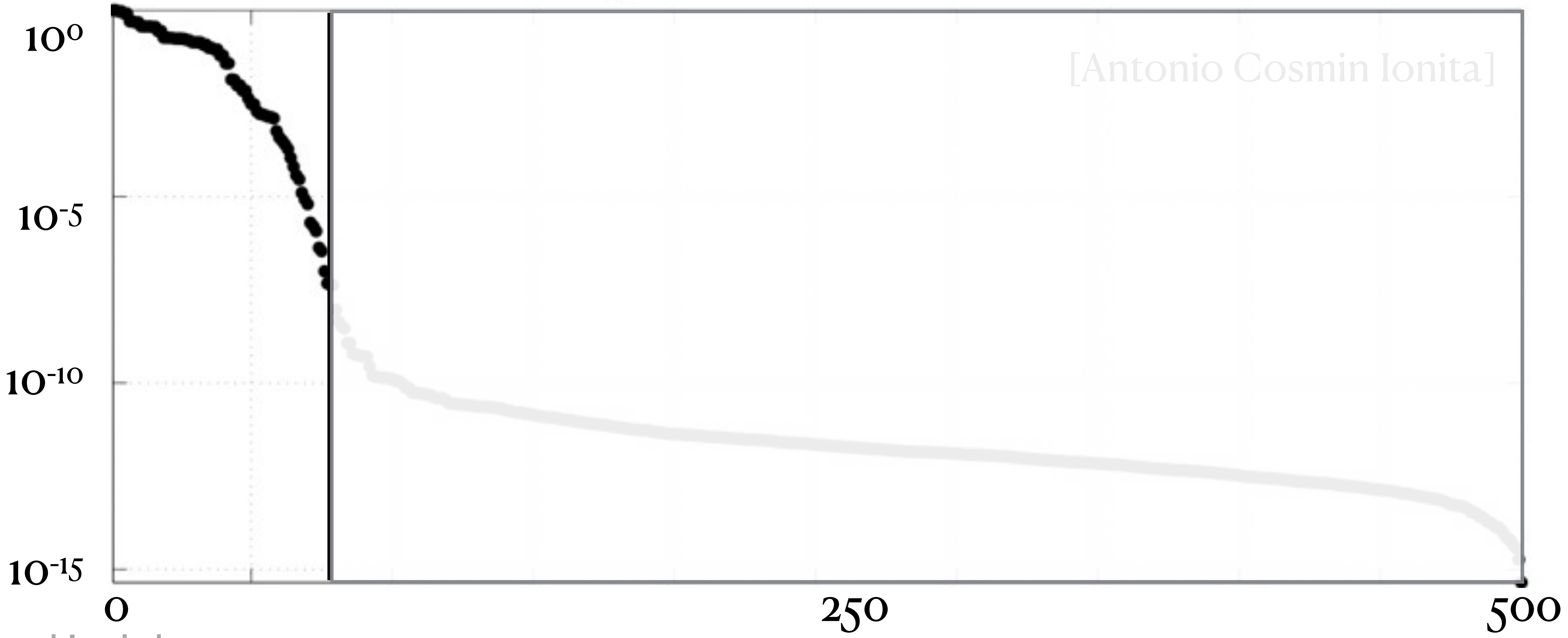
Plot of singular values (in decreasing magnitude)



Matrix approximation

$$\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \longrightarrow \mathbf{A} \approx \sum_{i=1}^{n_{\max}} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Plot of singular values (in decreasing magnitude)



Eckart-Young Theorem

Stated without proof.

Suppose that **A'** is obtained from **A** by truncating all but the largest *k* singular values from its singular value decomposition.

Then **A'** minimizes both

$$\text{(i)} \quad \|\mathbf{A} - \mathbf{A}'\|_F \quad \textit{and}$$

$$\text{(ii)} \quad \|\mathbf{A} - \mathbf{A}'\|_2$$

among all matrices **A'** with rank *k*.

Eckart-Young Theorem

Stated without proof.

Suppose that **A'** is obtained from **A** by truncating all but the largest *k* singular values from its singular value decomposition.

Then **A'** minimizes both

(i) $||\mathbf{A} - \mathbf{A}'||_F$ *and*

(ii) $||\mathbf{A} - \mathbf{A}'||_2$

$$||\mathbf{A}||_F^2 := \sum_{ij} A_{ij}^2$$

among all matrices **A'** with rank *k*.

("Frobenius norm")

Demo time

Computing inverses via the SVD

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Computing inverses via the SVD

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

$$\mathbf{A}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T$$

Computing inverses via the SVD

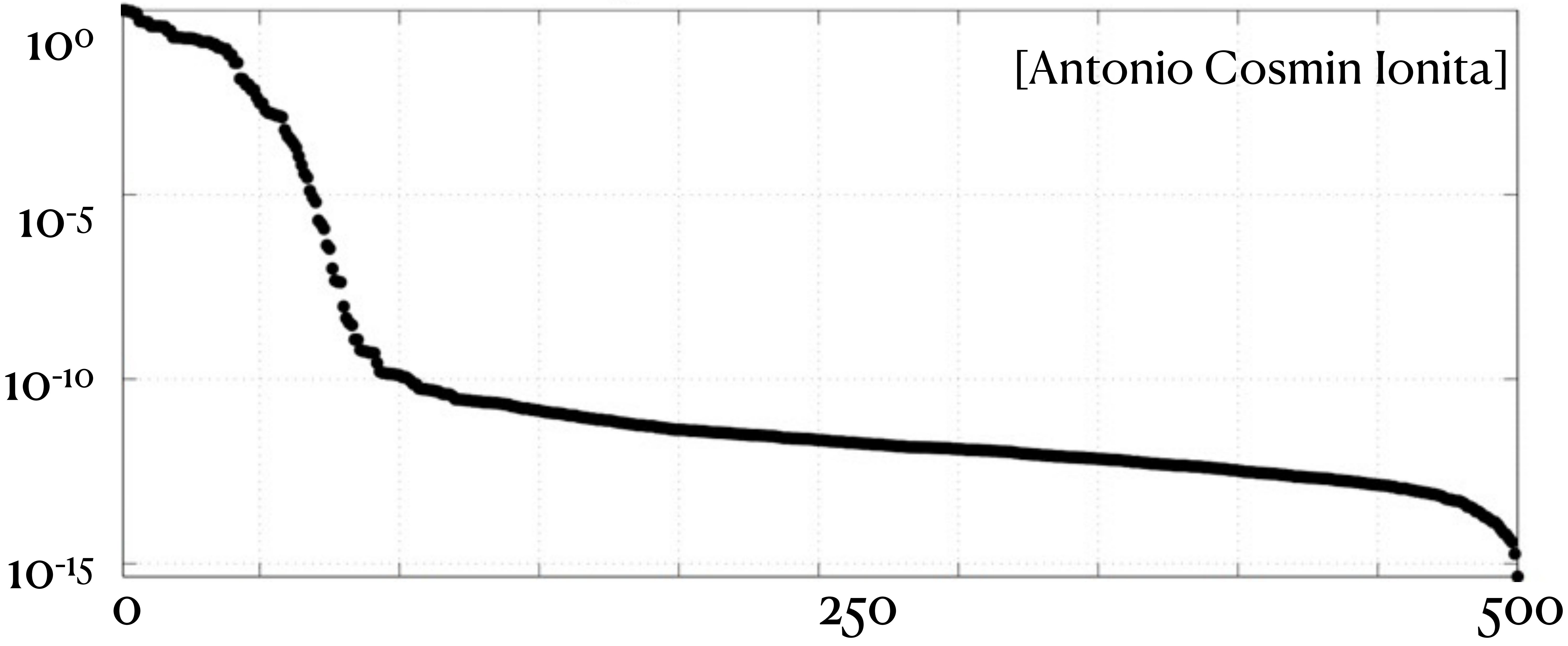
$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

$$\mathbf{A}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T = \sum_{i=1}^{n} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T$$

Computing inverses via the SVD

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

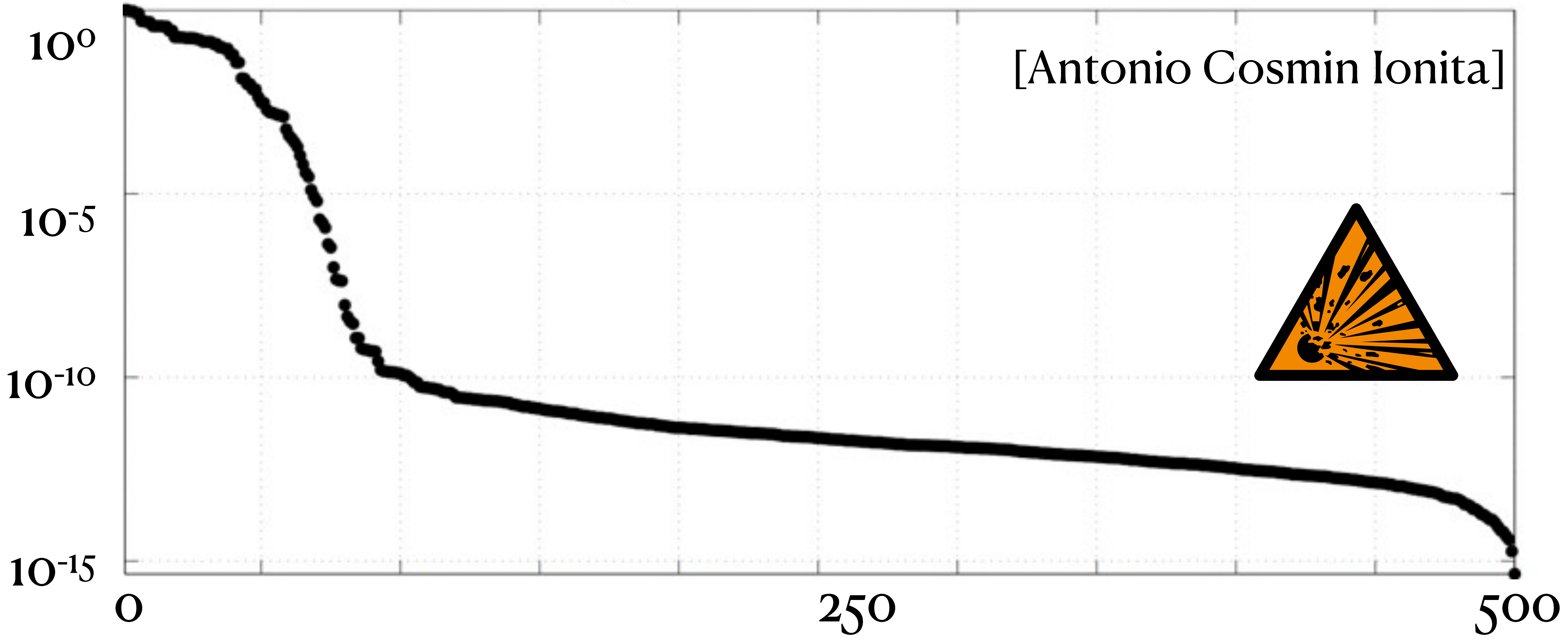
Plot of singular values (in decreasing magnitude)



Computing inverses via the SVD

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

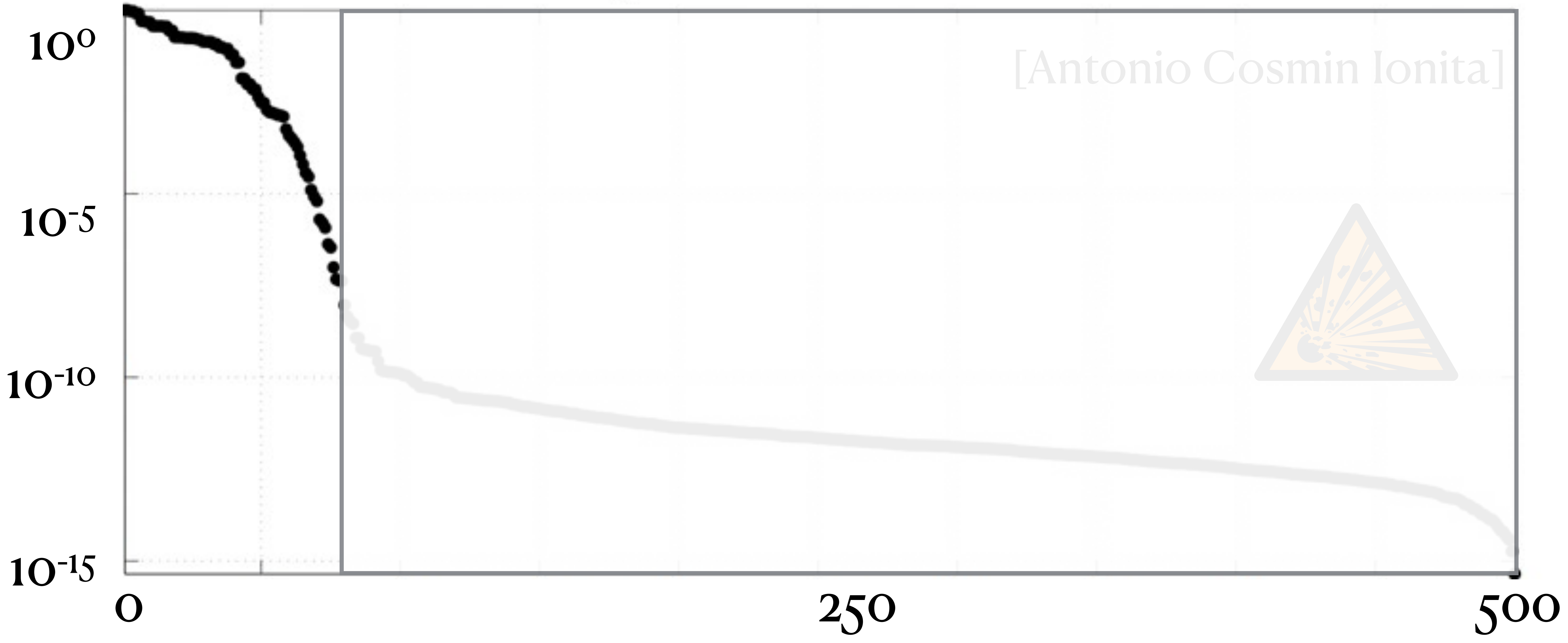
Plot of singular values (in decreasing magnitude)



Computing inverses via the SVD

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

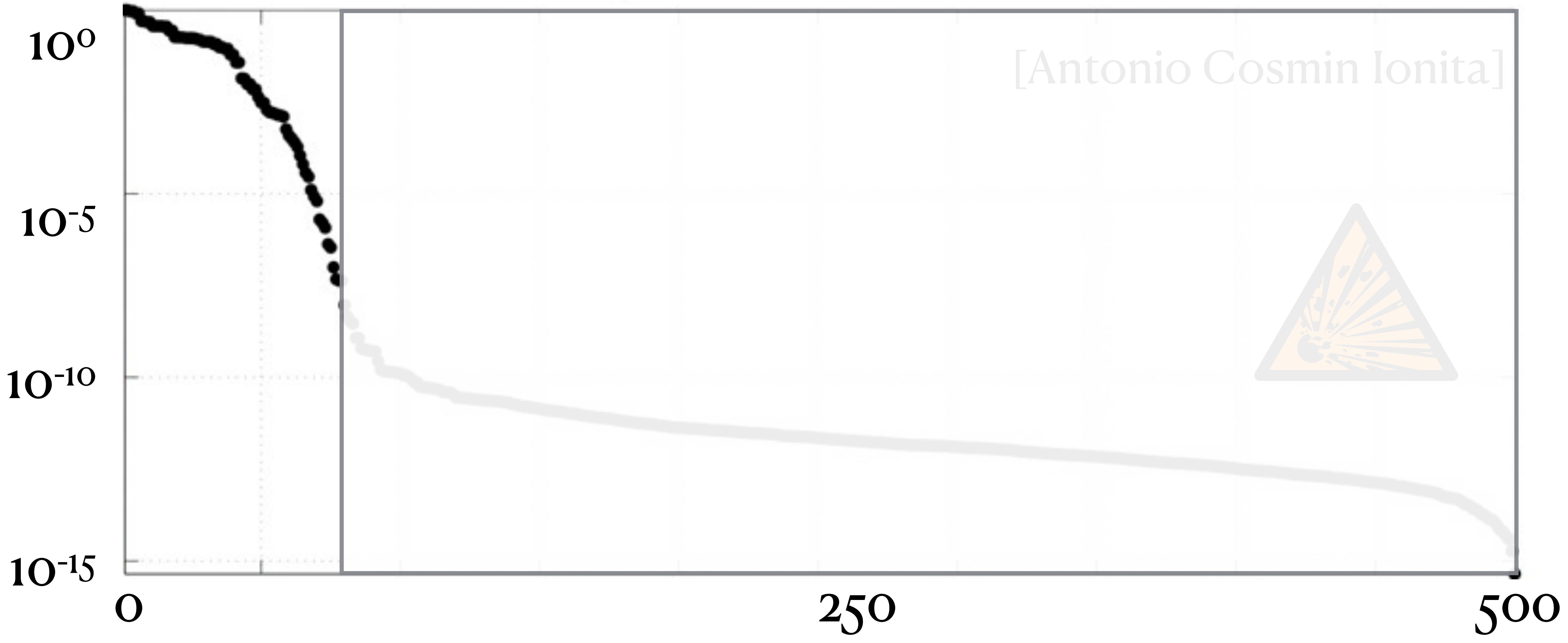
Plot of singular values (in decreasing magnitude)



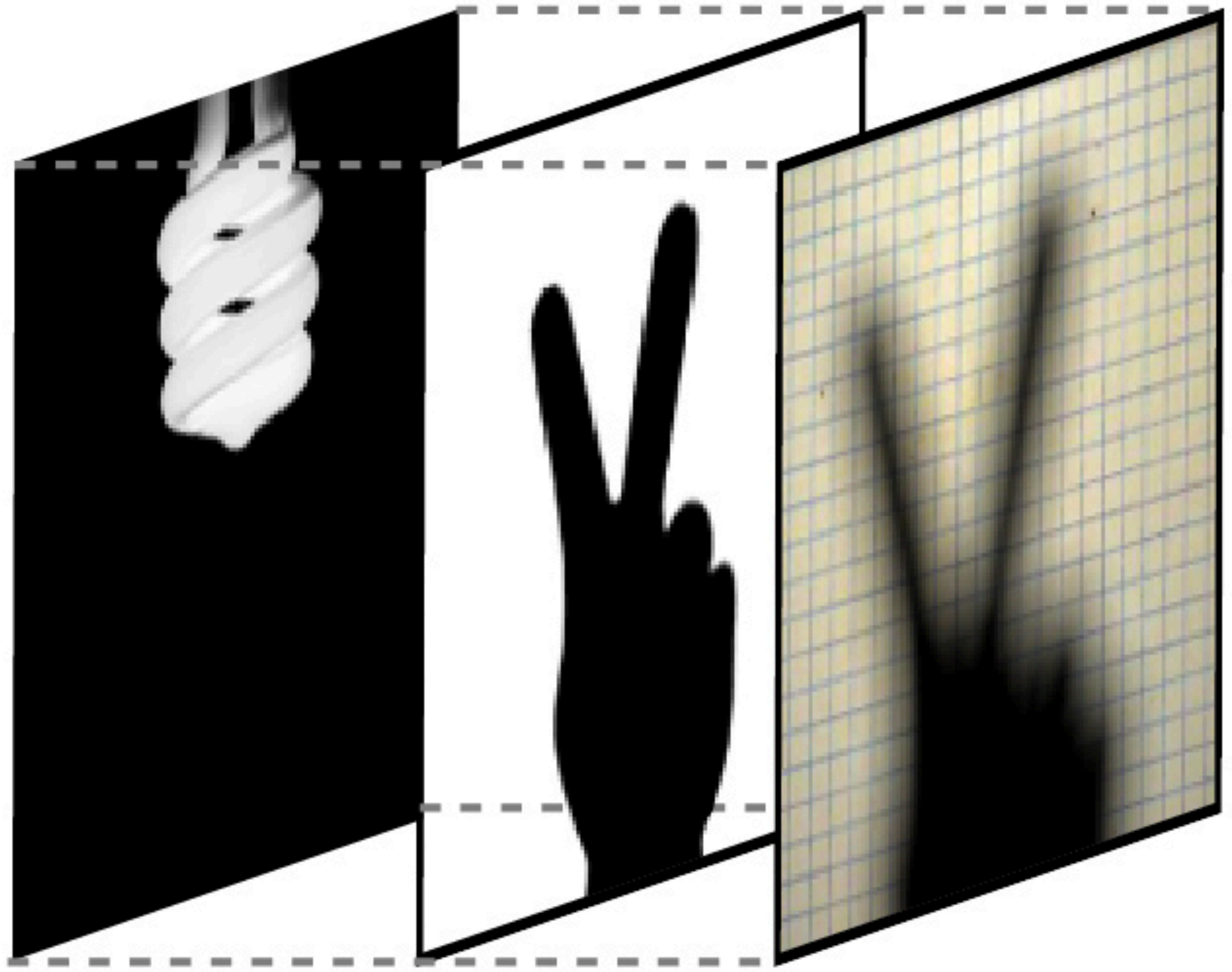
Computing inverses via the SVD

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T \approx \sum_{i=1}^{n_{\max}} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

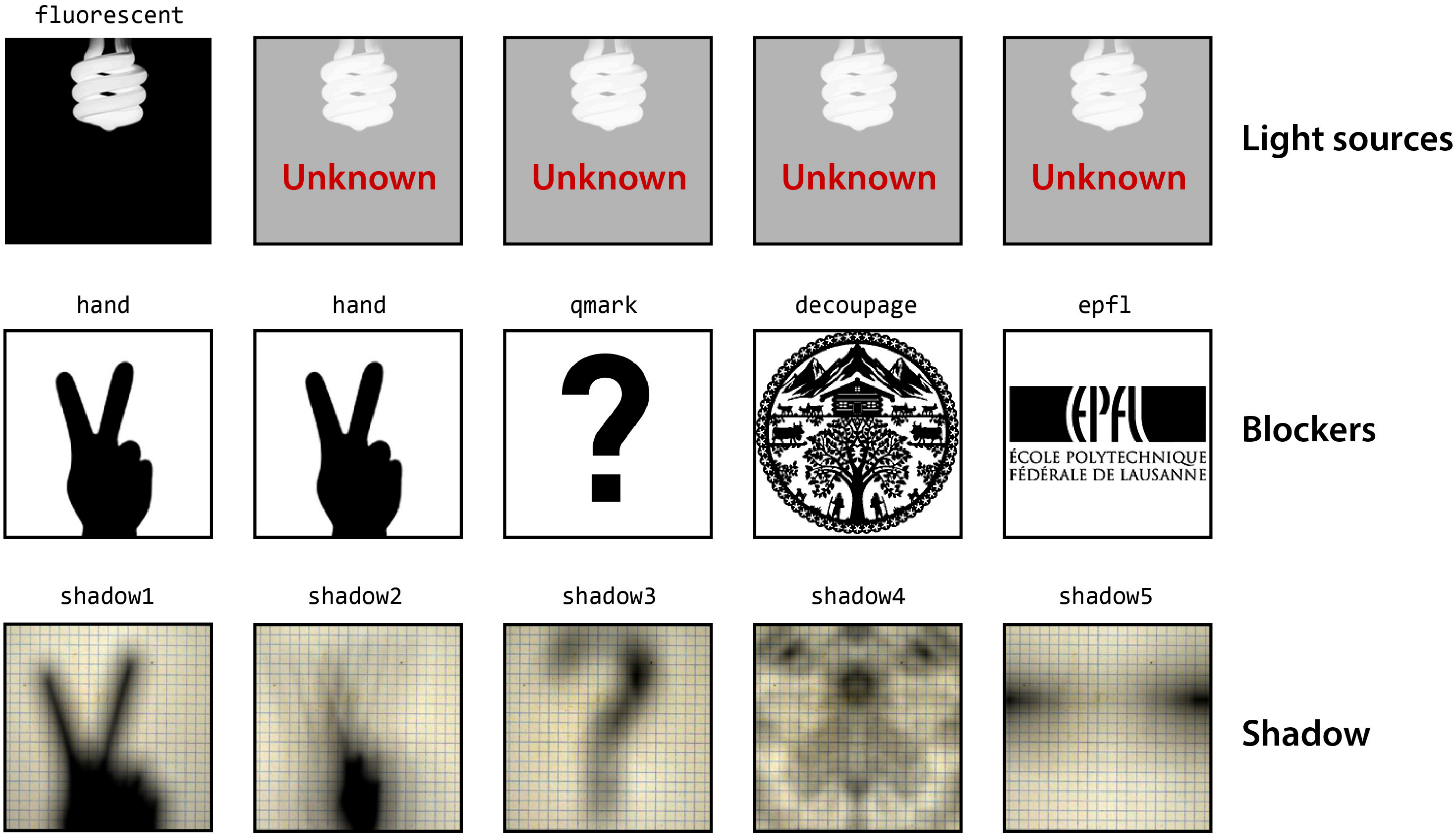
Plot of singular values (in decreasing magnitude)



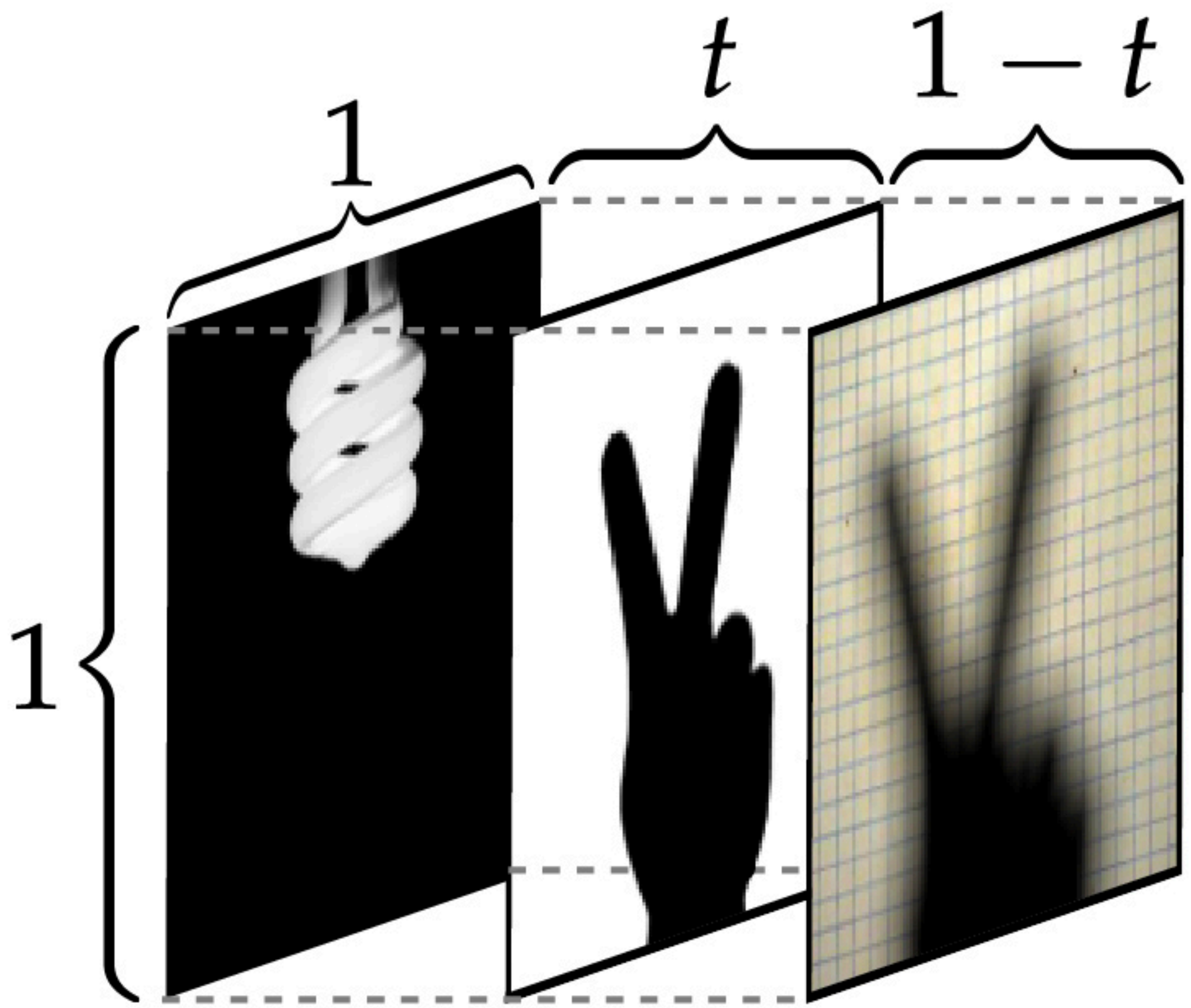
SVD as a regularization strategy



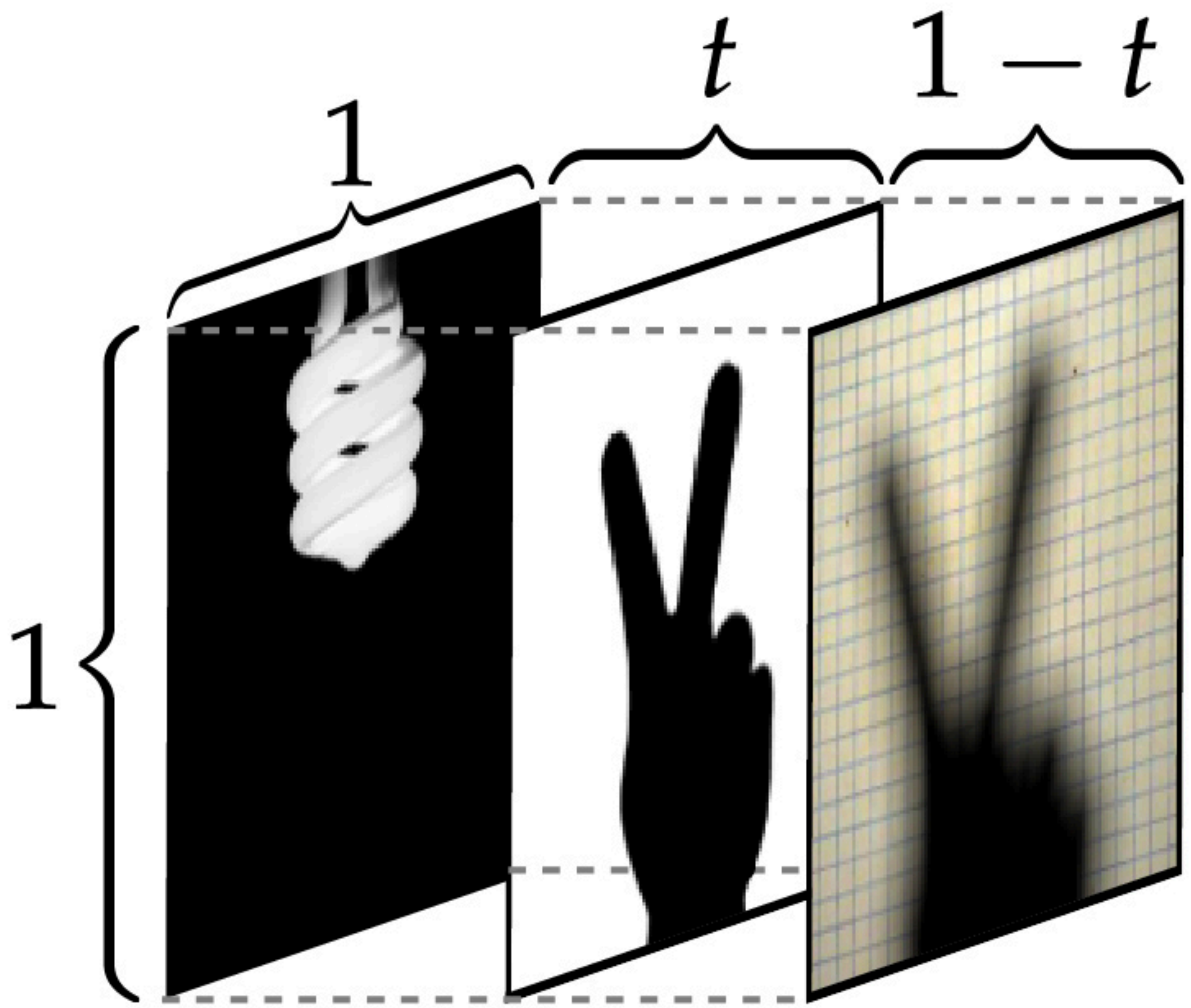
The *SVD-powered X-ray glasses*
(originally by Doug James @ Stanford U.)

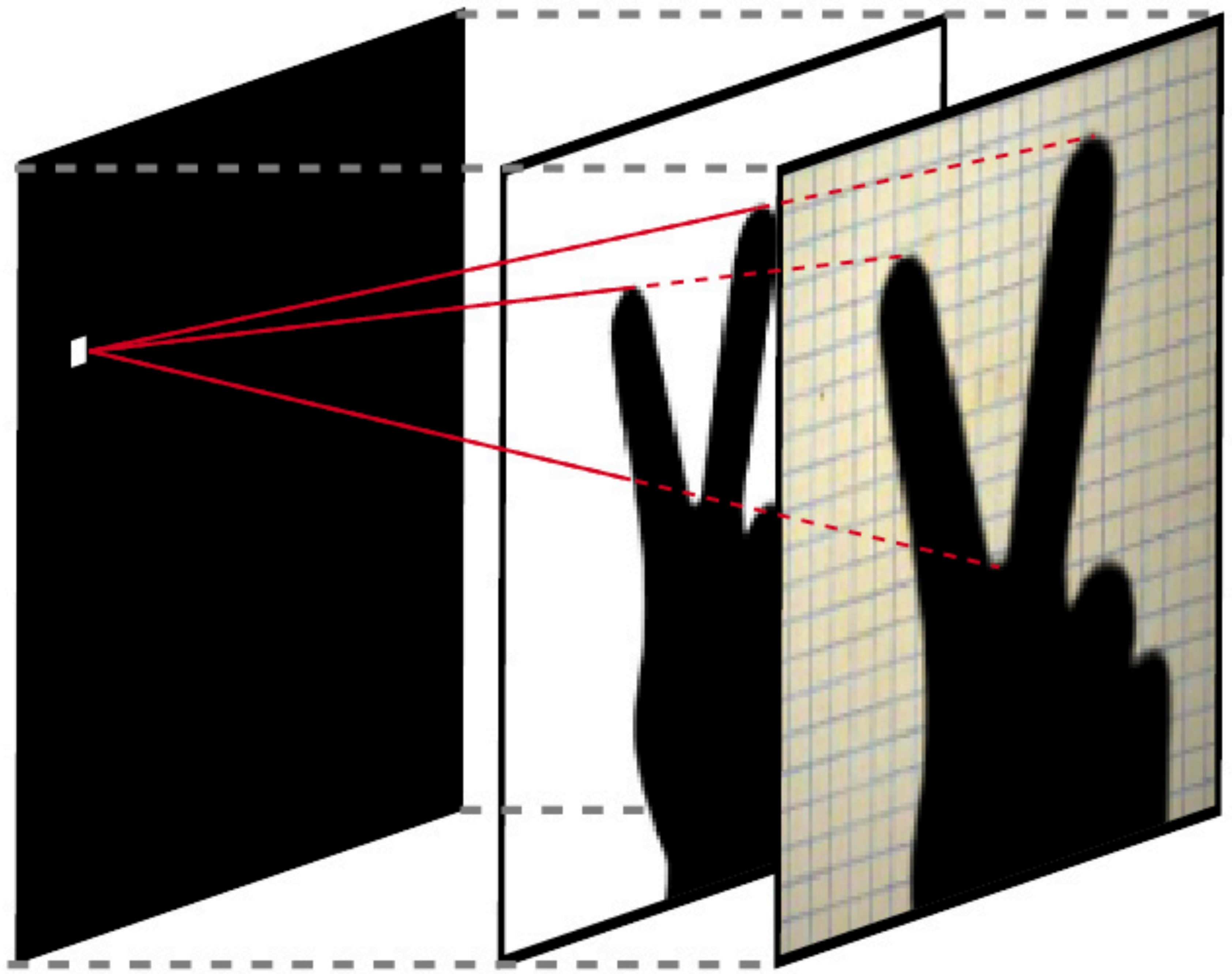

fluorescent
Unknown
Unknown
Unknown
Unknown
Light sources
hand
hand
qmark
decoupage
epfl
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Blockers
shadow1
shadow2
shadow3
shadow4
shadow5
Shadow

Problem setup

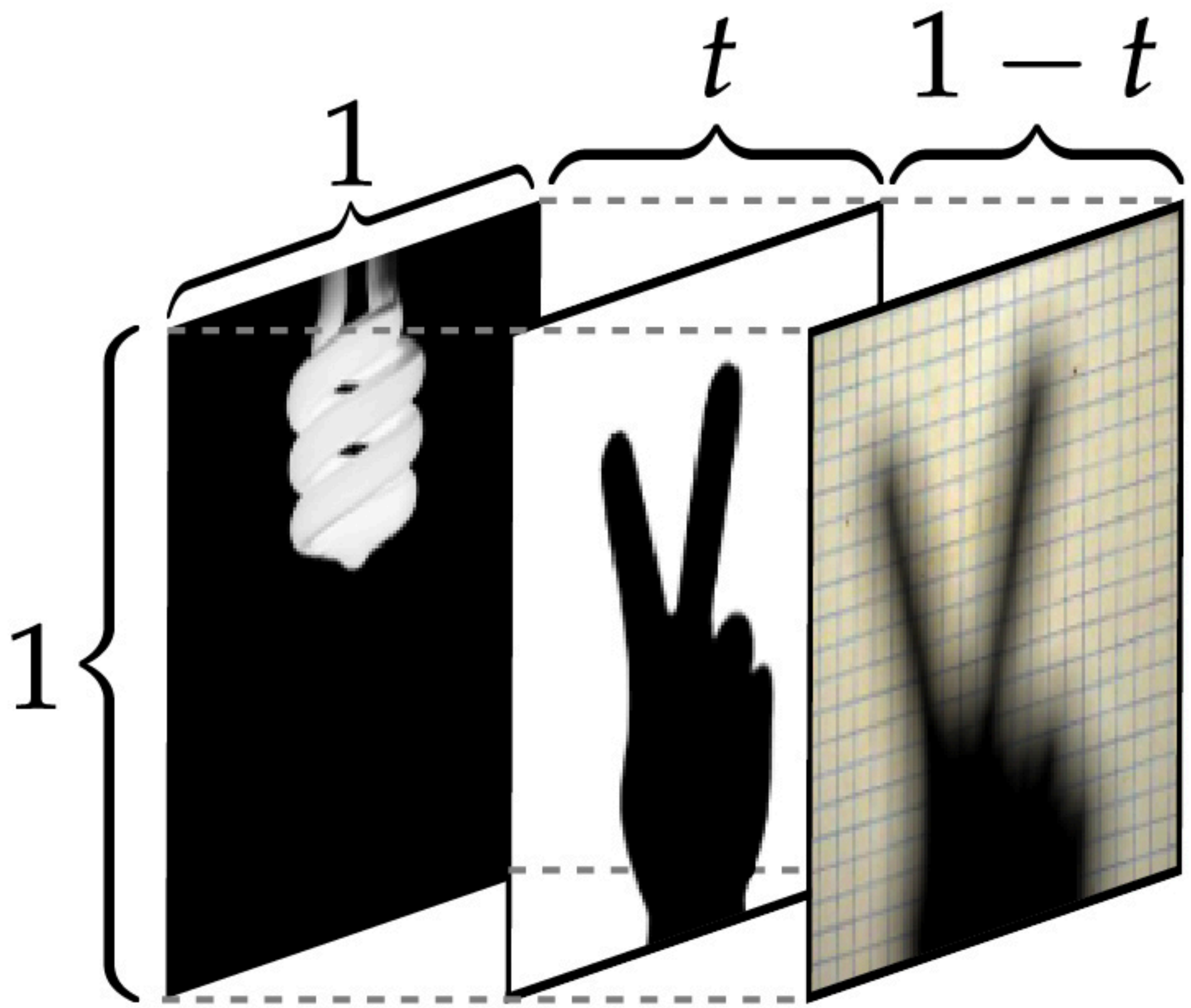


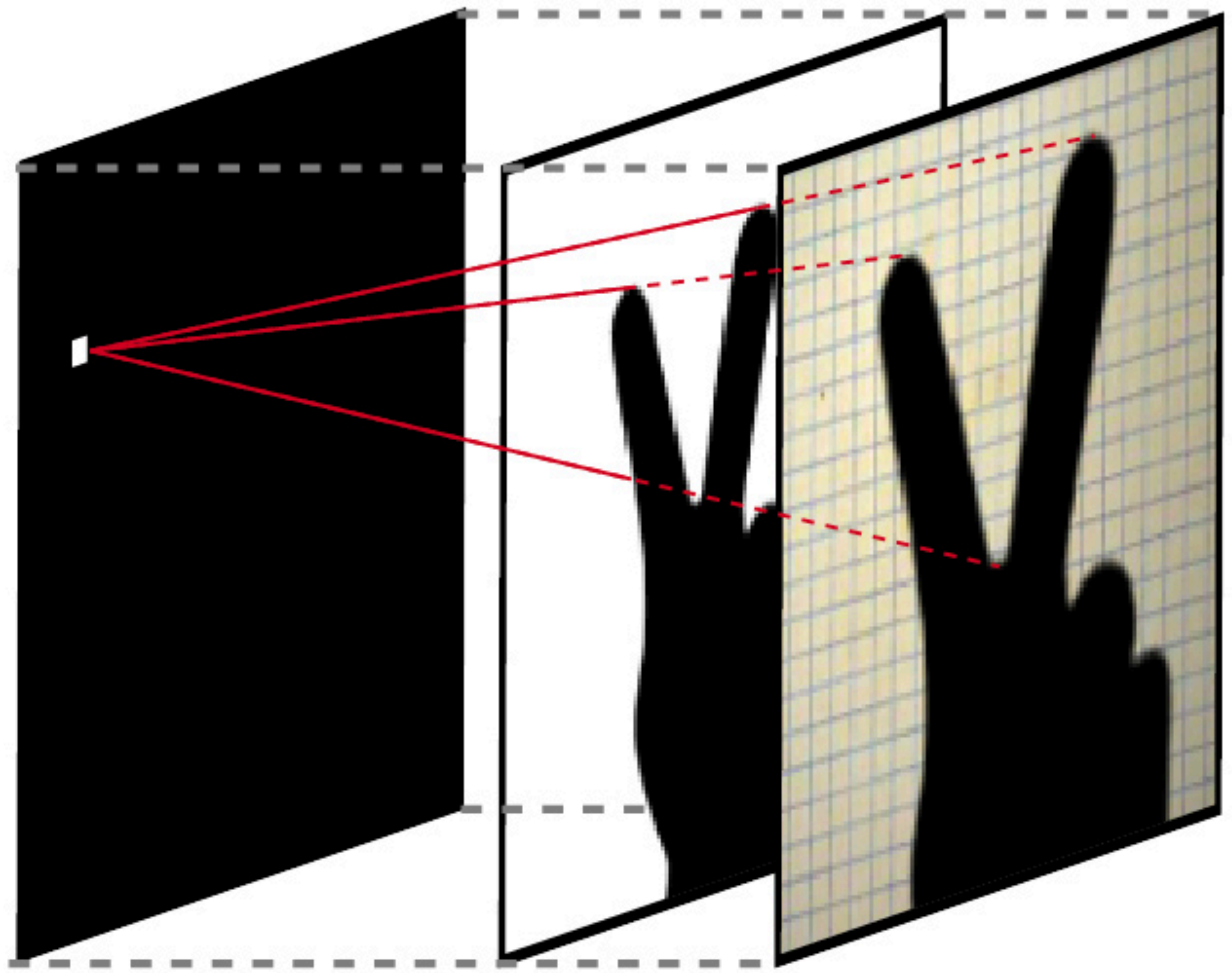
Problem setup

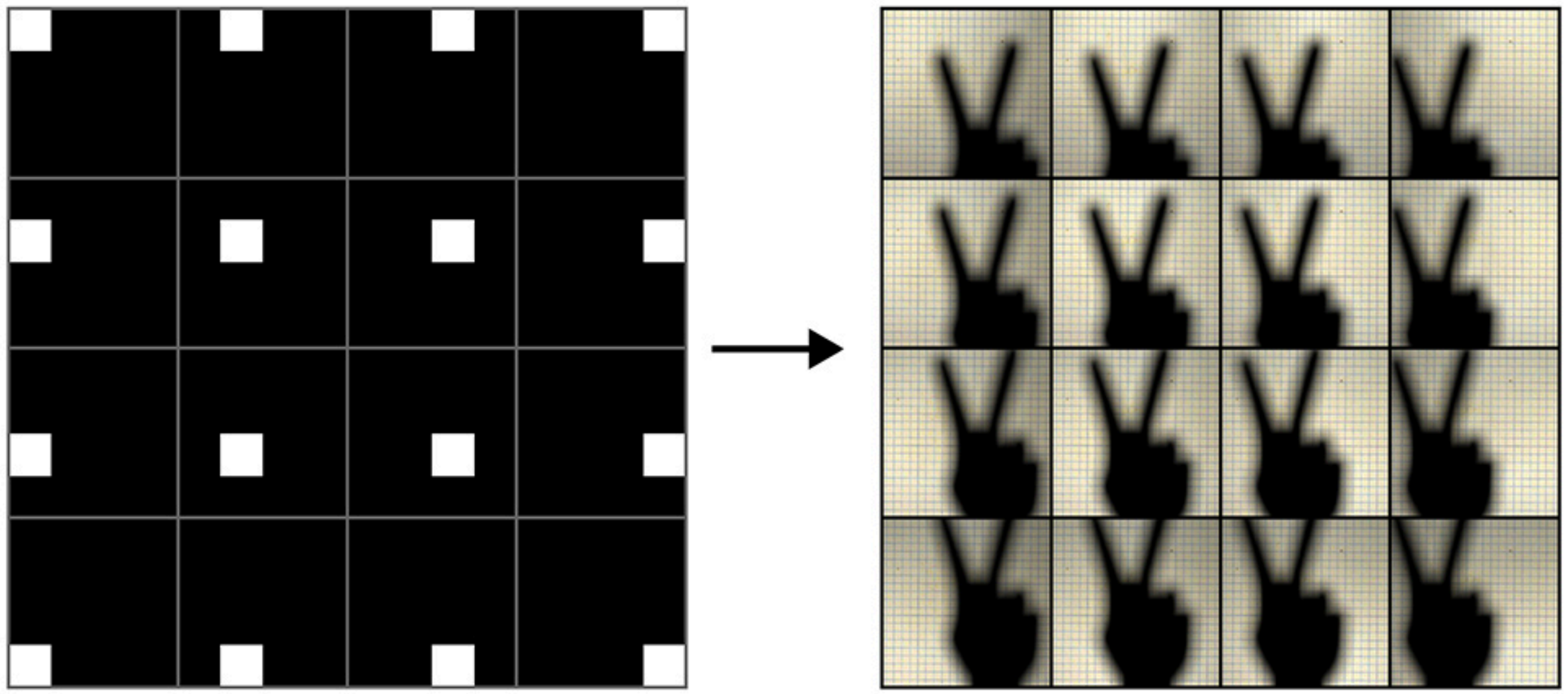




Problem setup



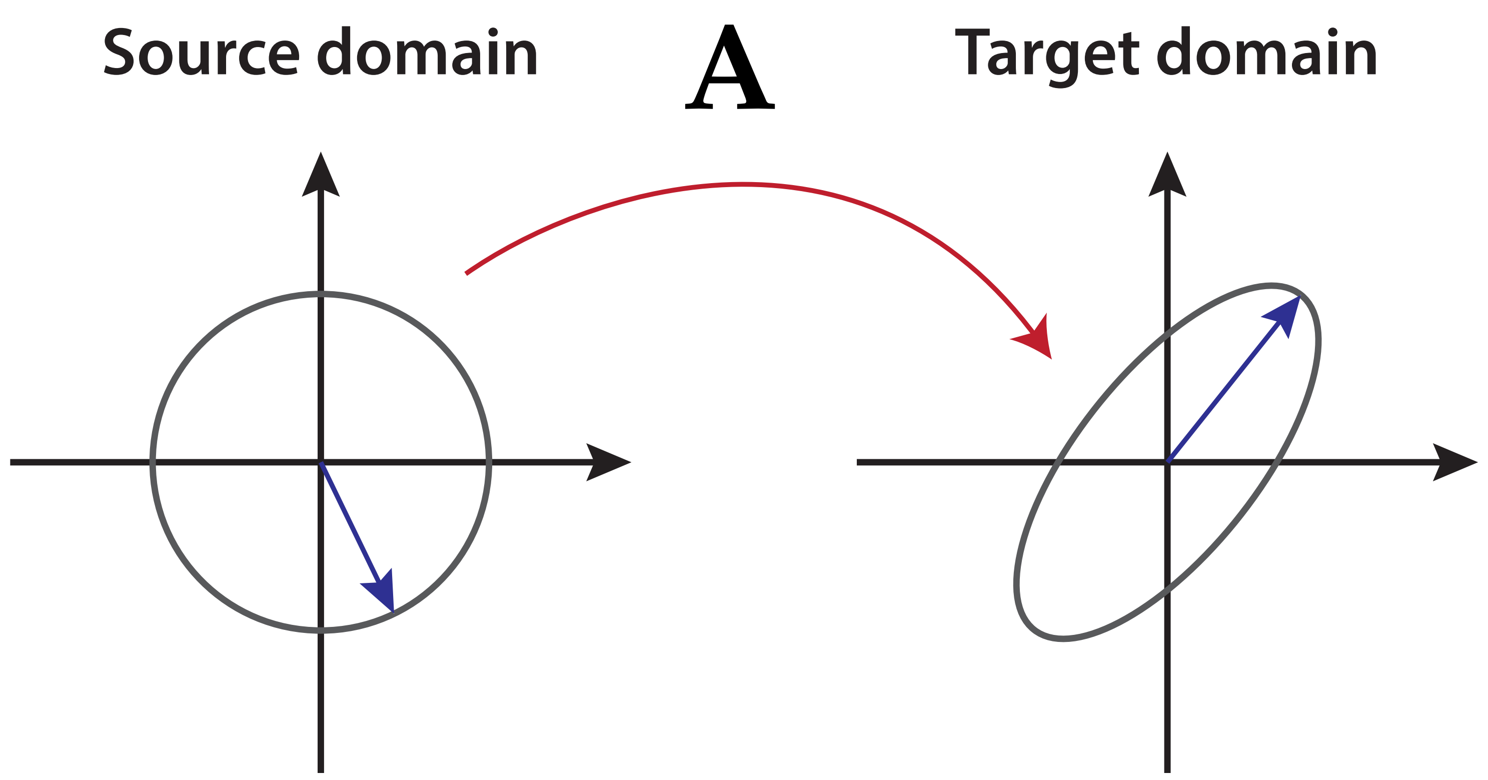


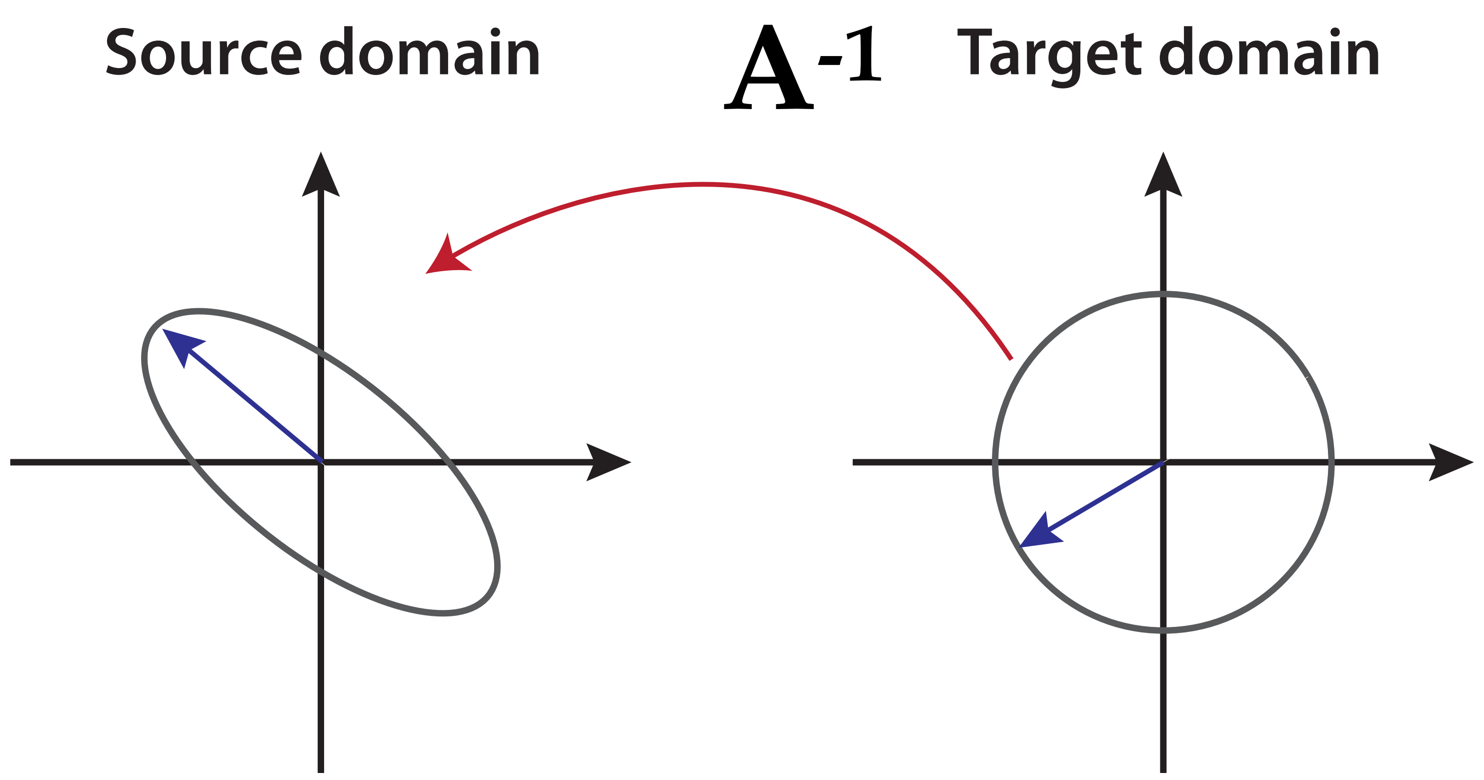


Demo time

Revisiting the condition number

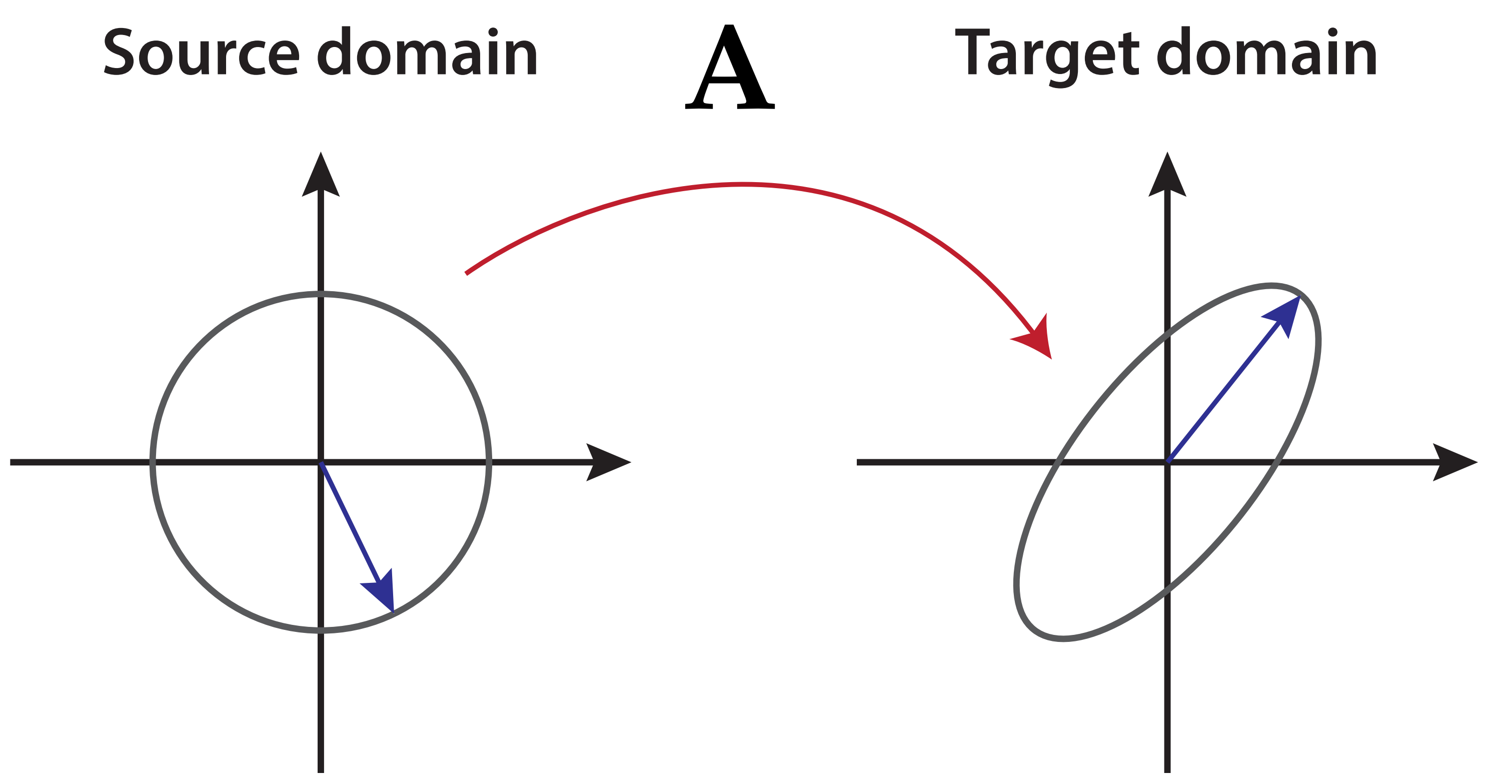
$$\operatorname{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

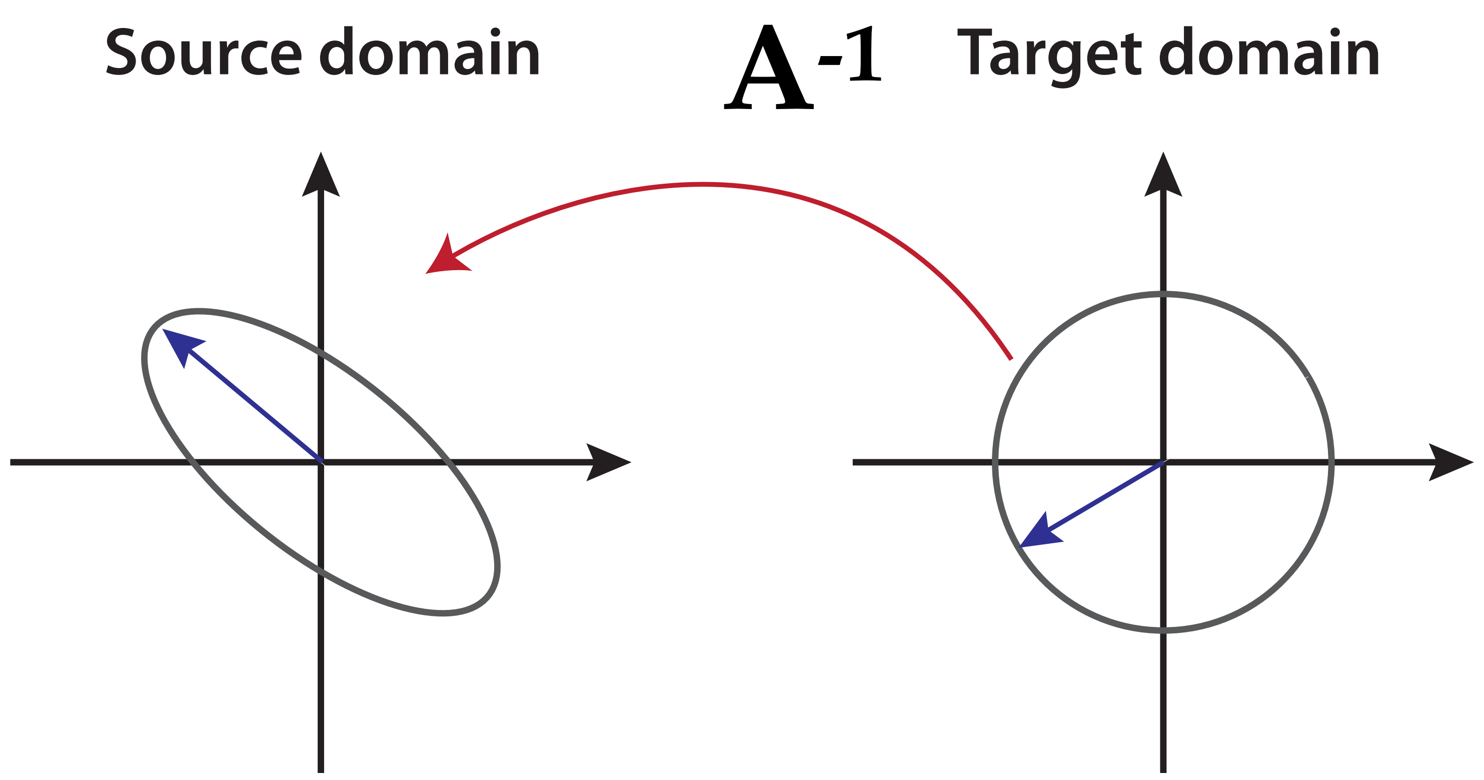




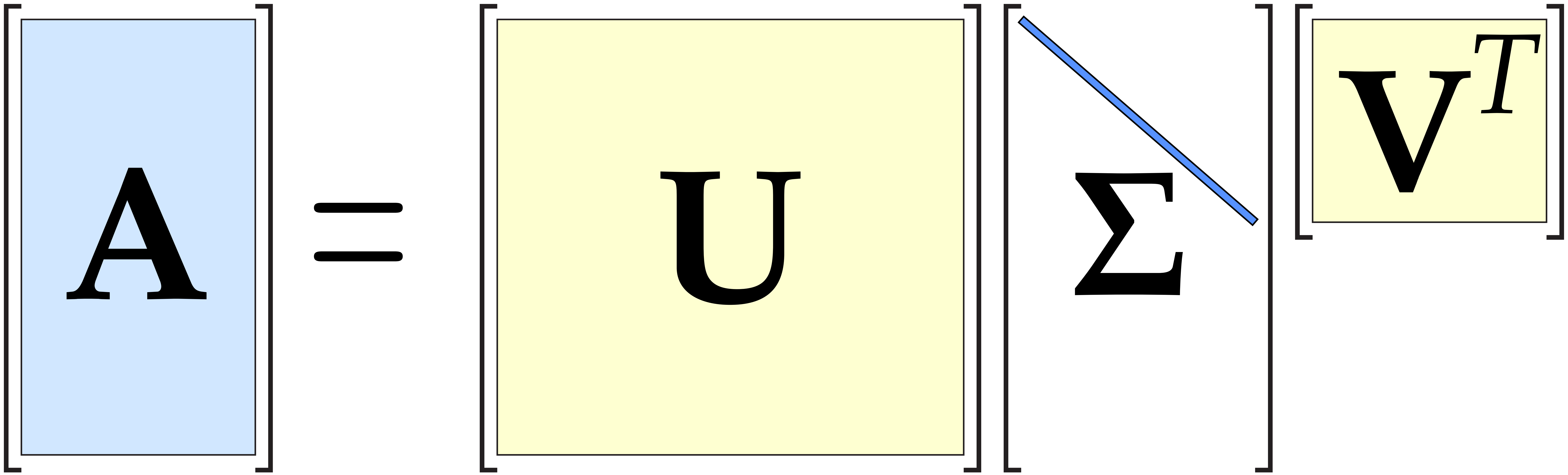
Revisiting the condition number

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$
$$= \frac{\sigma_1}{\sigma_n}.$$

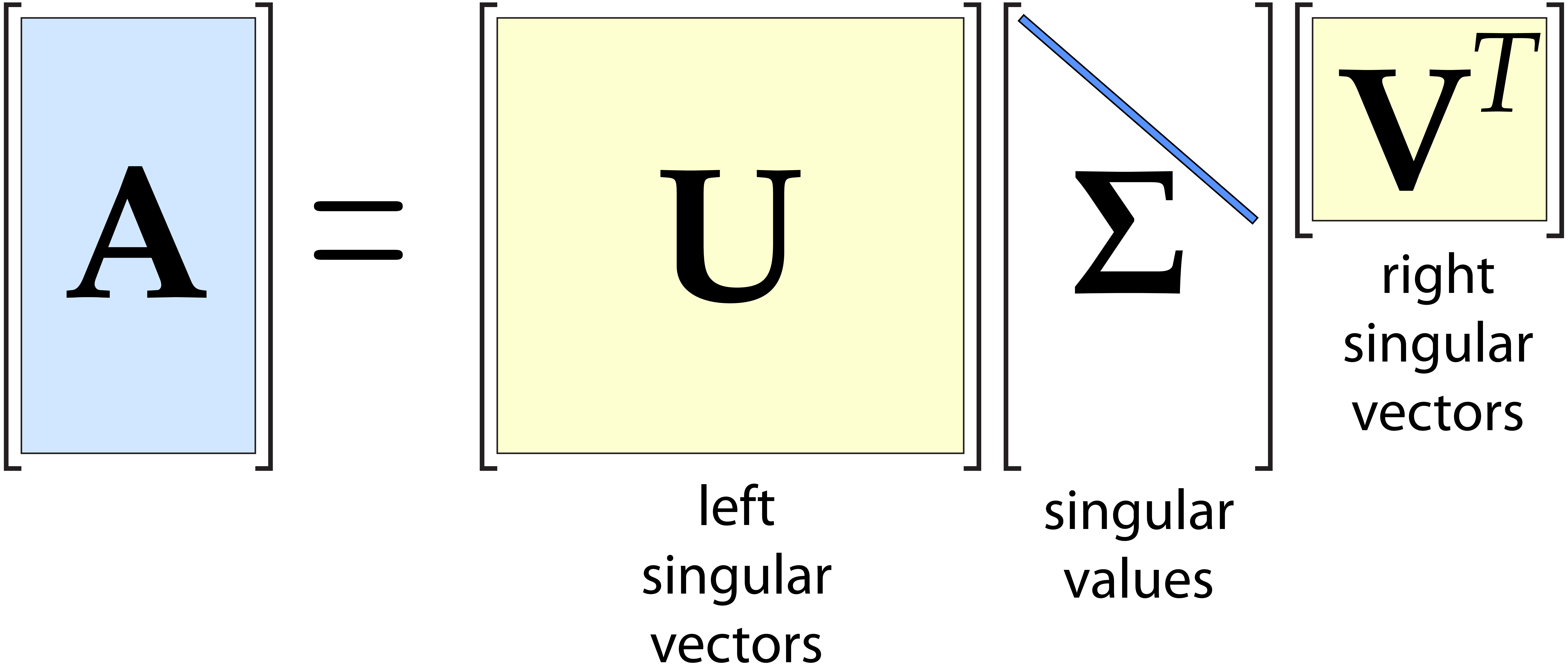




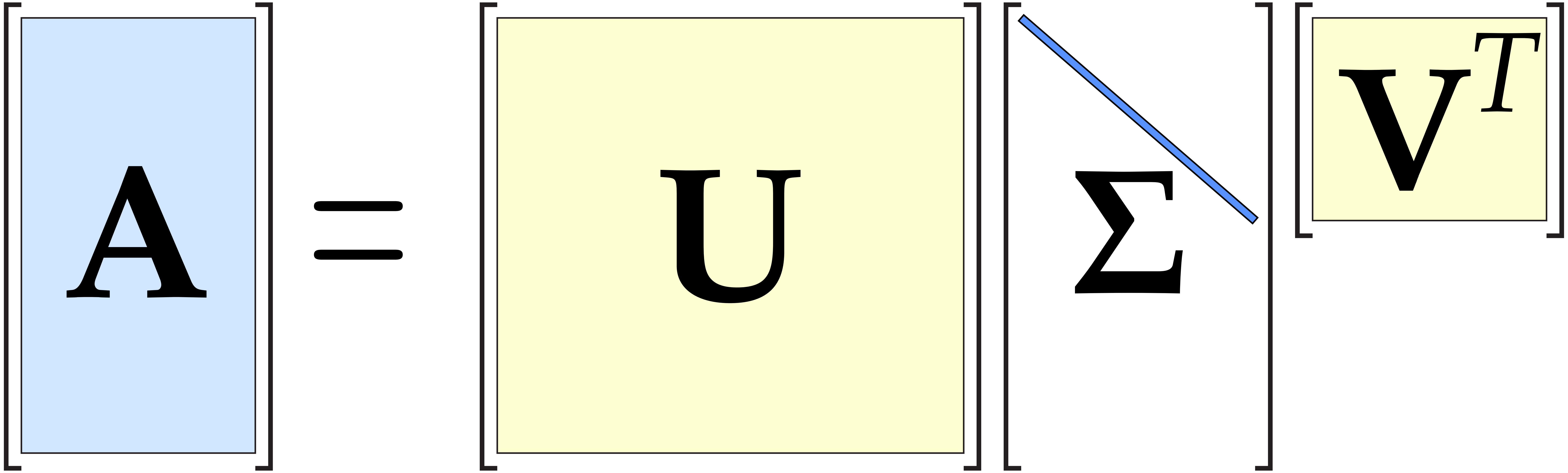
SVD shape (tall & wide case)



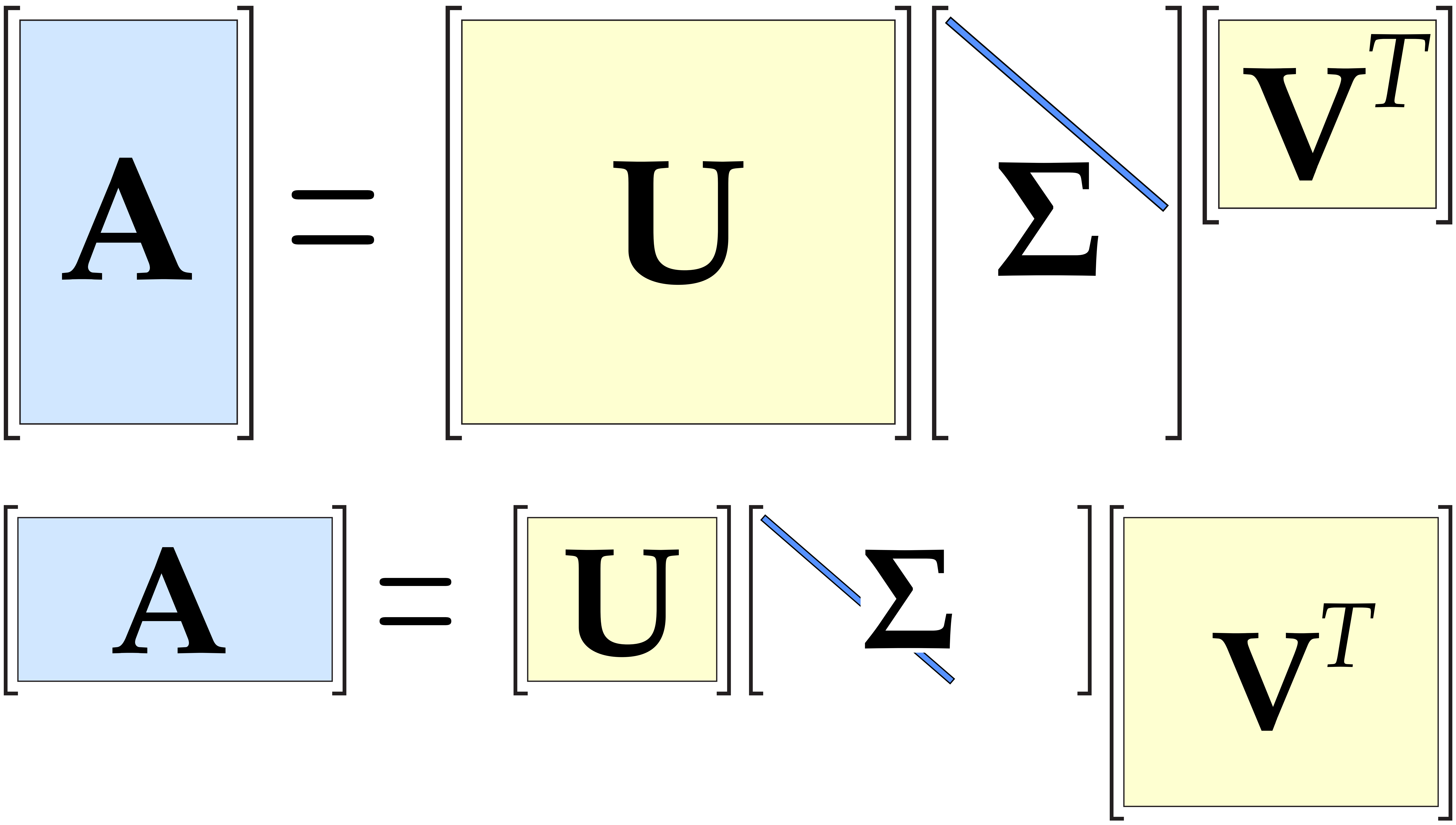
SVD shape (tall & wide case)



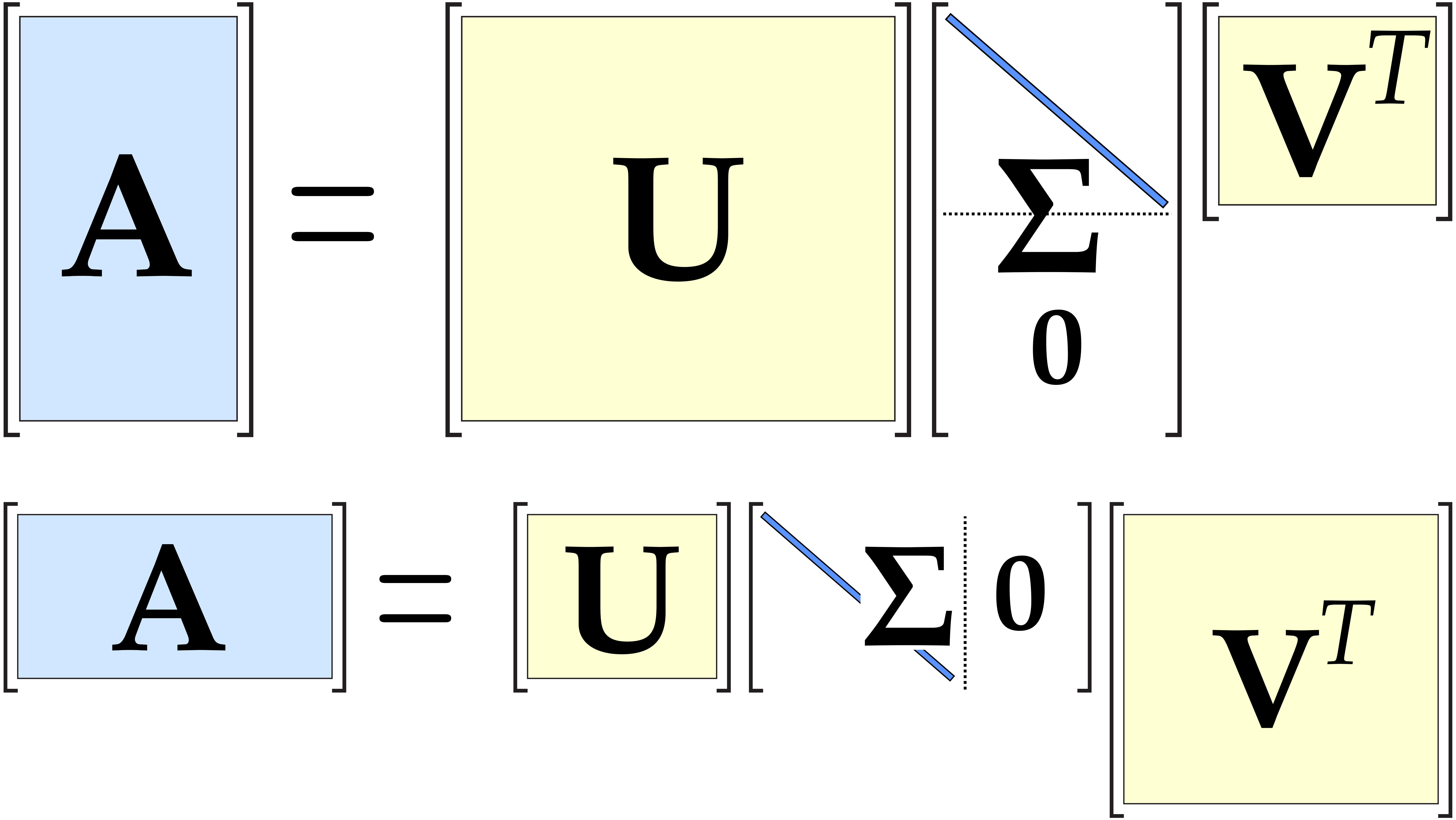
SVD shape (tall & wide case)



SVD shape (tall & wide case)



SVD shape (tall & wide case)



The Pseudoinverse

SVD provides the "ultimate" form of a matrix inverse

$$\mathbf{A}^+ = \sum_{i=1}^{\min\{n,m\}} \mathbf{v}_i \mathbf{u}_i^T \begin{cases} \frac{1}{\sigma_i}, & \sigma_i \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

The Pseudoinverse

SVD provides the "ultimate" form of a matrix inverse

$$\mathbf{A}^+ = \sum_{i=1}^{\min\{n,m\}} \mathbf{v}_i \mathbf{u}_i^T \begin{cases} \frac{1}{\sigma_i}, & \sigma_i \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

1. Identical to inverse for square & full-rank matrices.

The Pseudoinverse

SVD provides the "ultimate" form of a matrix inverse

$$\mathbf{A}^{+} = \sum_{i=1}^{\min\{n,m\}} \mathbf{v}_i \mathbf{u}_i^T \begin{cases} \frac{1}{\sigma_i}, & \sigma_i \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

1. Identical to inverse for square & full-rank matrices.
2. Gives least-squares solution for overconstrained / tall linear systems

The Pseudoinverse

SVD provides the "ultimate" form of a matrix inverse

$$\mathbf{A}^+ = \sum_{i=1}^{\min\{n,m\}} \mathbf{v}_i \mathbf{u}_i^T \begin{cases} \frac{1}{\sigma_i}, & \sigma_i \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

1. Identical to inverse for square & full-rank matrices.
2. Gives least-squares solution for overconstrained / tall linear systems
3. Gives minimum-norm solution for underconstrained /wide linear systems