Linear algebra

Session 02: Inner product, solving systems of linear equations

Gerhard Jäger

November 2, 2023

The inner product of vectors

- the **inner product** of two vectors is a scalar

$$
\begin{aligned}
\mathbf{x} \cdot \mathbf{y} &\doteq x_1 y_1 + x_2 y_2 + \cdots + x_n y_n \\
&= \sum_i x_i y_i
\end{aligned}
$$

(Sometimes the inner product is written $\langle \mathbf{x}, \mathbf{y} \rangle$.)

- the inner product is commutative

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x}$$

- Furthermore, the inner product is **linear** in both arguments

$$
\begin{aligned}
(a\mathbf{x}) \cdot (b\mathbf{y}) &= ab(\mathbf{x} \cdot \mathbf{y}) \\
\mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) &= \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z} \\
(\mathbf{x} + \mathbf{y}) \cdot \mathbf{z} &= \mathbf{x} \cdot \mathbf{z} + \mathbf{y} \cdot \mathbf{z}
\end{aligned}
$$

norm of a vector

The **norm** (=length) of a vector is defined as

$$
\begin{aligned}
\|\mathbf{x}\| &\doteq \sqrt{\mathbf{x} \cdot \mathbf{x}} \\
&= \sqrt{\sum_i x_i^2}
\end{aligned}
$$

properties of the norm

- for all vectors $\mathbf{x}, \mathbf{y}$ and scalars a:

$$
\begin{aligned}
\|\mathbf{x}\| &\geq 0 \\
\|\mathbf{x}\| &= 0 \text{ if and only if } \mathbf{x} = \mathbf{0}\ (\forall i.\, x_i = 0) \\
\|a\mathbf{x}\| &= a\|\mathbf{x}\| \\
\|\mathbf{x} + \mathbf{y}\| &\leq \|\mathbf{x}\| + \|\mathbf{y}\|
\end{aligned}
$$

unit vectors

A **unit vector** is a vector of length 1.

Examples:

Out[4]:
$$\begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}$$

Out[5]:
$$\begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ \frac{2}{3} \end{bmatrix}$$

Out[6]:
$$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

Out[7]: $\begin{bmatrix} \sin(x) \\ \cos(x) \end{bmatrix}$

Out[8]: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

We can always shrink or stretch a vector into a unit vector of the same direction by dividing it by its norm.

$$\frac{\mathbf{u}}{||\mathbf{u}||} \text{ is always a unit vector, provided } \mathbf{u} \neq \mathbf{0}.$$

angle between vectors

For unit vectors, the dot product has a simple geometric interpretation:

If

$$||\mathbf{u}|| = ||\mathbf{v}|| = 1,$$

then

$$\mathbf{u} \cdot \mathbf{v} = \cos\theta,$$

where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$.

Proof: [https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product (https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product)](https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product)

From this it follows

$$\begin{aligned} \mathbf{u} &= ||\mathbf{u}|| \frac{\mathbf{u}}{||\mathbf{u}||} \\ \mathbf{v} &= ||\mathbf{v}|| \frac{\mathbf{v}}{||\mathbf{v}||} \\ \mathbf{u} \cdot \mathbf{v} &= (||\mathbf{u}|| \frac{\mathbf{u}}{||\mathbf{u}||}) \cdot (||\mathbf{v}|| \frac{\mathbf{v}}{||\mathbf{v}||}) \\ &= ||\mathbf{u}|| \, ||\mathbf{v}|| (\frac{\mathbf{u}}{||\mathbf{u}||} \cdot \frac{\mathbf{v}}{||\mathbf{v}||}) \\ &= ||\mathbf{u}|| \, ||\mathbf{v}|| \cos\theta \\ \cos\theta &= \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||} \end{aligned}$$

This is how the cosine is defined in analytical geometry. (Note that this only holds if $\mathbf{u} \neq \mathbf{0}, \mathbf{v} \neq \mathbf{0}$.)

Since the cosine is always ≤ 1, it follows **("Scharz Inequality")**:

$$\mathbf{u} \cdot \mathbf{v} \leq ||\mathbf{u}|| \, ||\mathbf{v}||$$

triangle inequality

$$\begin{aligned} ||\mathbf{u} + \mathbf{v}|| &= \sqrt{(\mathbf{u} + \mathbf{v}) \cdot (\mathbf{u} + \mathbf{v})} \\ &= \sqrt{\mathbf{u} \cdot \mathbf{u} + 2\mathbf{u} \cdot \mathbf{v} + \mathbf{v} \cdot \mathbf{v}} \\ &= \sqrt{||\mathbf{u}||^2 + 2||\mathbf{u}|| \, ||\mathbf{v}|| \cos\theta + ||\mathbf{v}||^2} \\ &\leq \sqrt{||\mathbf{u}||^2 + 2||\mathbf{u}|| \, ||\mathbf{v}|| + ||\mathbf{v}||^2} \\ &\leq ||\mathbf{u}|| + ||\mathbf{v}|| \end{aligned}$$

orthogonal vectors

The inner product of two vectors can be 0. Examples

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix}, \begin{bmatrix} -8 \\ 6 \end{bmatrix}$$

Matrices

A $m \times n$ matrix is a sequence of m row-vectors, each of length n or, equivalently, a sequence of n column vectors, each of length m.

Example of a 3×2 matrix:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Individual cells are referred to with two indices (first: row, second: column), often using the lowercase version of the name of the matrix.

$$\begin{aligned} a_{1,1} &= 1 \\ a_{3,2} &= 6 \\ &\vdots \end{aligned}$$

The **transpose** of a matrix (written with T as an exponent) is the result of flipping rows and columns.

$$A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Obviously, the transpose of a $m \times n$ matrix is an $n \times m$ matrix.

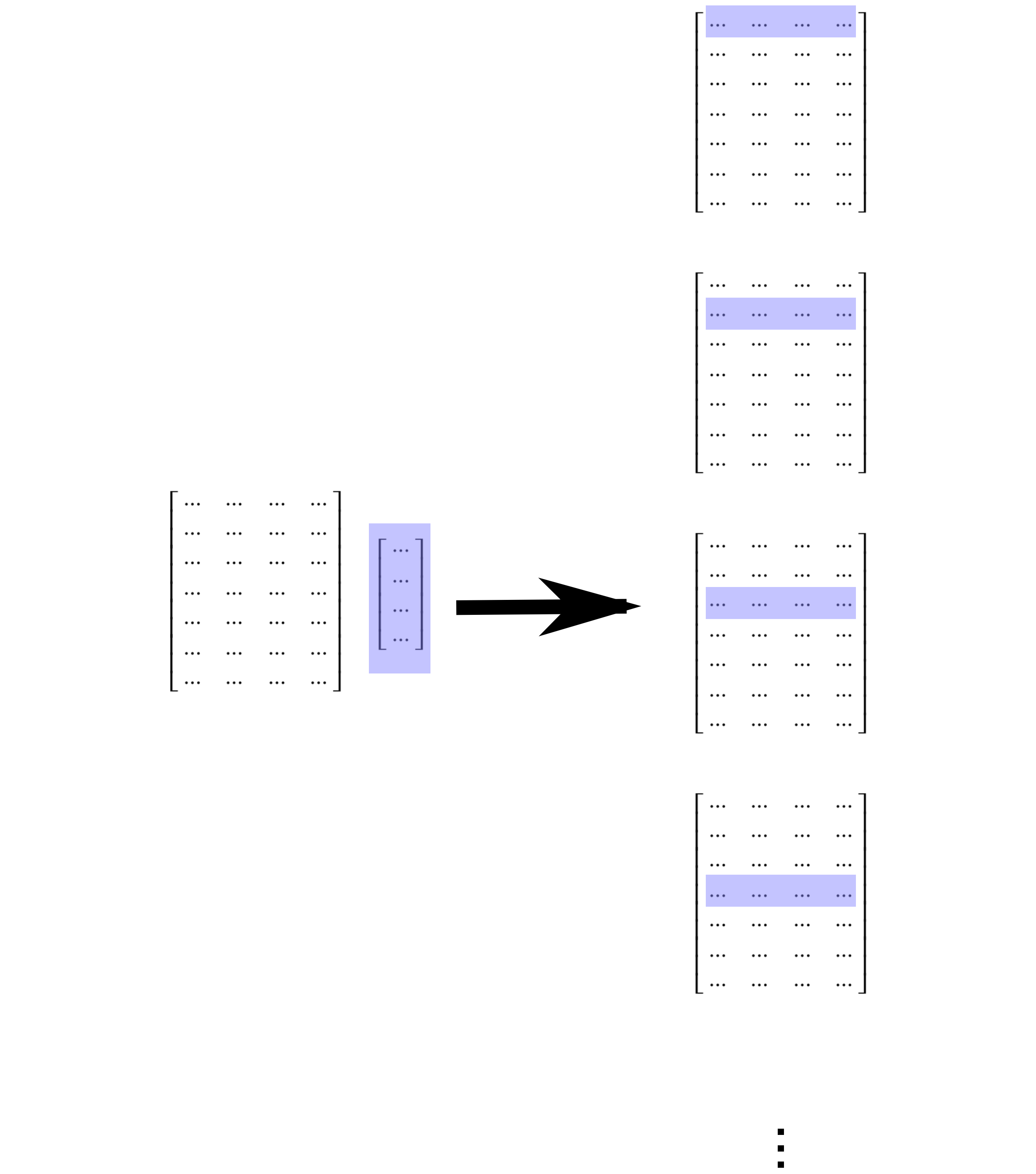
Matrix operations

Applying a matrix to a vector

A $m \times n$ matrix can be seen as a function from $\mathbb{R}^n$ into $\mathbb{m}$. (Note that the number of columns reflects the input size and the number of rows the output size.)

General definition

$$A\mathbf{x} = \begin{bmatrix} \sum_{1 \le i \le n} a_{1,i} x_i \\ \sum_{1 \le i \le n} a_{2,i} x_i \\ \vdots \\ \sum_{1 \le i \le n} a_{m,i} x_i \end{bmatrix} = \begin{bmatrix} A_{1,\_} \cdot \mathbf{x} \\ A_{2,\_} \cdot \mathbf{x} \\ \vdots \\ A_{m,\_} \cdot \mathbf{x} \end{bmatrix}$$



Example

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Column picture

So far we focused on rows. Equivalently, this can be conceived as a column operation:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

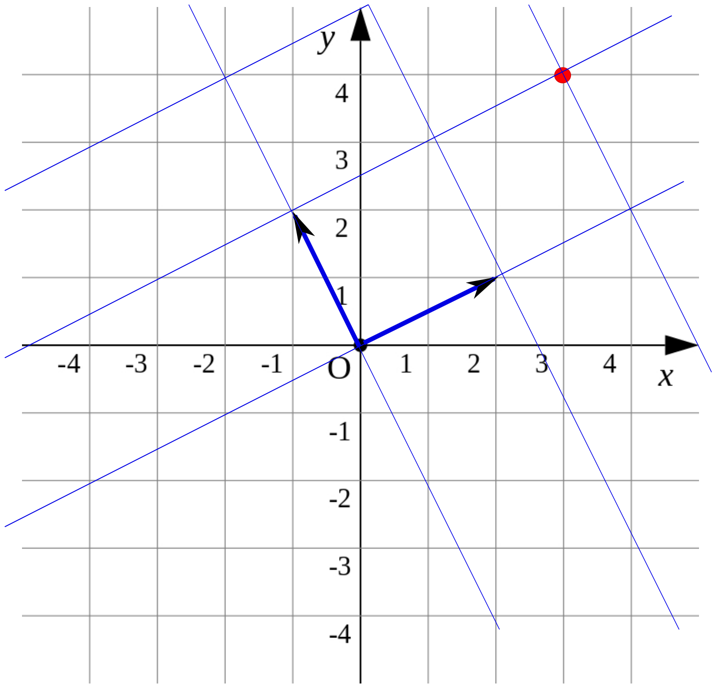
When computing $A\mathbf{x}$, each column of A can be seen as the axis of some (possibly squewed or degenerate) **coordinate system**. Applying A to $\mathbf{x}$ means:

- $\mathbf{x}$ is a vector in the coordinate system defined by the columns of A
- $A\mathbf{x}$ is the translation of $\mathbf{x}$ into the “objective” coordinate system.

$$A = \begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$A\mathbf{x} = 2\begin{bmatrix} 2 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

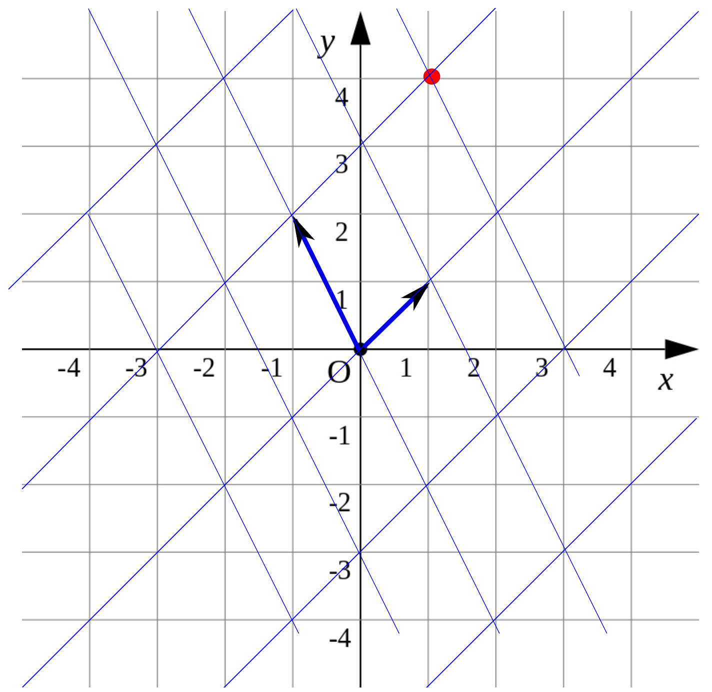


The columns of A need not be perpendicular.

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$A\mathbf{x} = 2\begin{bmatrix} 1 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

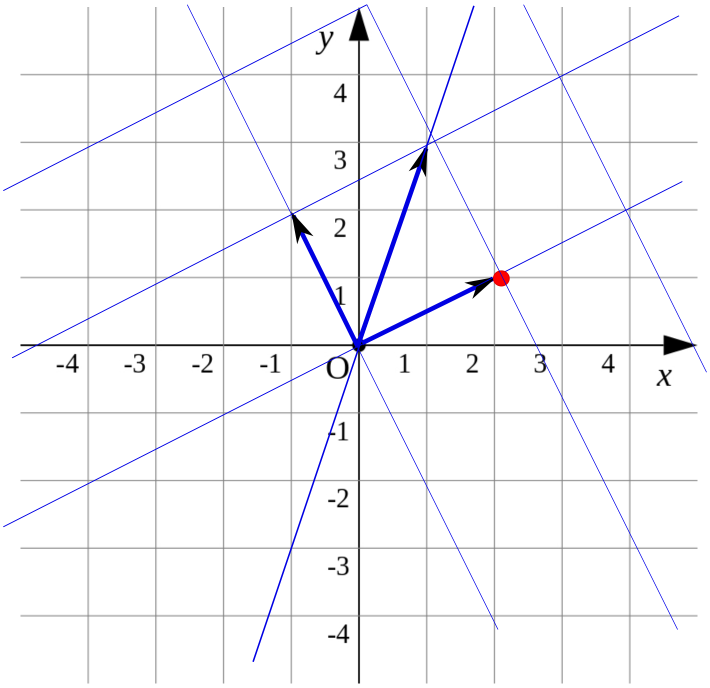


We can also have degenerate cases where the columns of A are not independent.

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$$

$$A\mathbf{x} = 2\begin{bmatrix} 2 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -1 \\ 2 \end{bmatrix} - 1\begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



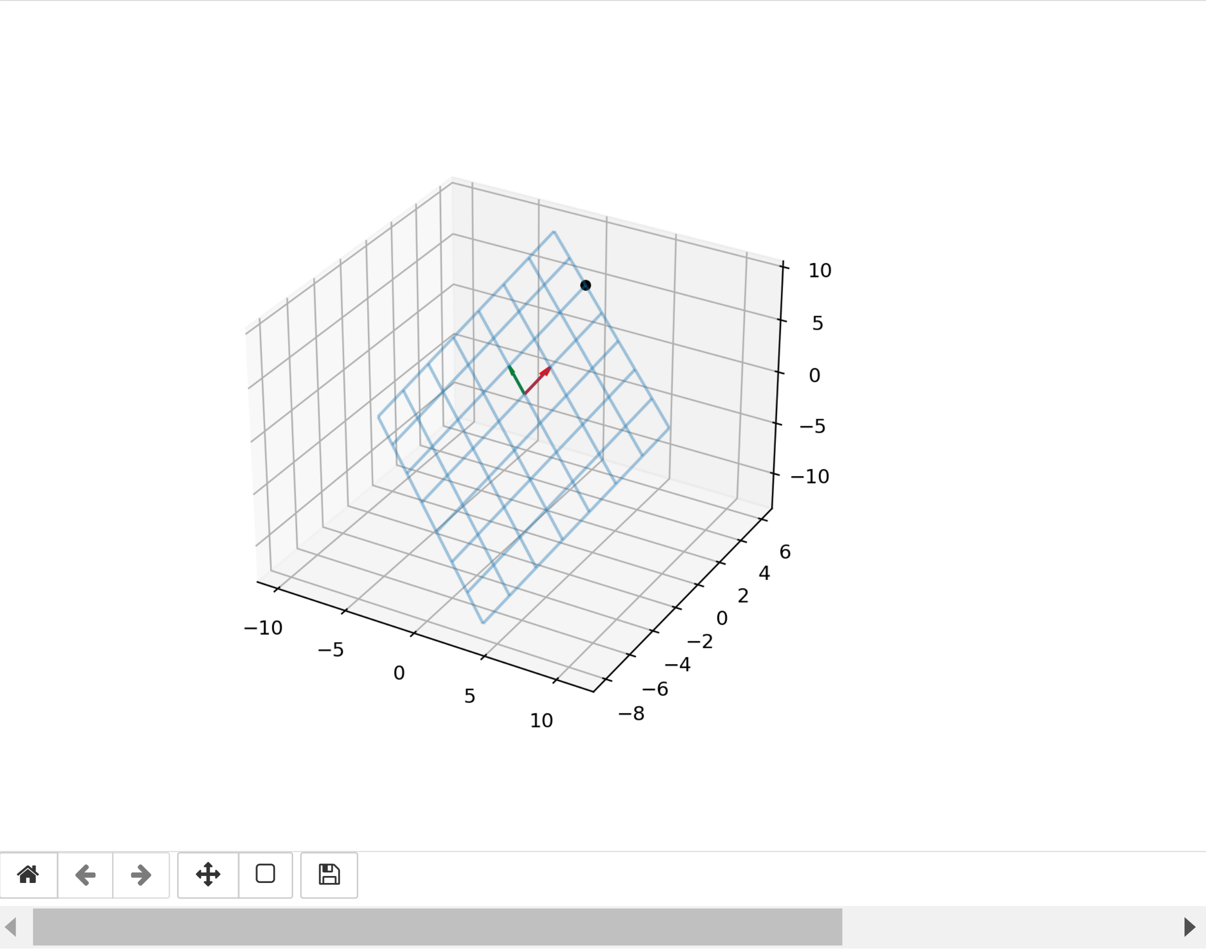
Conversely, A may project a low-dimensional vector into a higher-dimensional space.

$$A = \begin{bmatrix} 1 & -2 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$A\mathbf{x} = 3\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$$

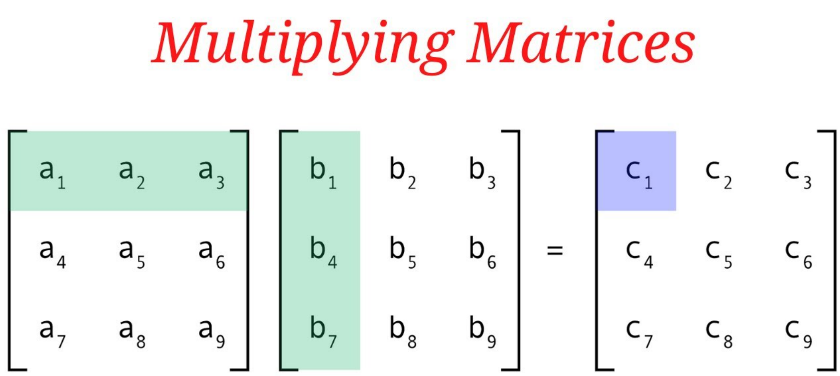
Figure 1



Matrix multiplication

If A is an $m \times n$ matrix and B is a $n \times o$ matrix, than the **matrix product** AB is an $m \times o$ matrix.

$$\begin{aligned}(AB)_{i,j} &= \sum_k a_{i,k} b_{kj} \\ &= A_{i,\_} \cdot B_{\_,j}\end{aligned}$$



properties of matrix multiplication

- matrix multiplication is **associative**

$$(AB)C = A(BC)$$

- matrix multiplication is **not commutative**. It is possible that

$$AB \neq BA$$

Example

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 5 & 6 & 4 \\ 8 & 9 & 7 \end{bmatrix}$$

special matrices

- a **diagonal matrix** is a matrix where all entries except the *main diagonal* (all $a_{i,i}$) are zero

Out[10]: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

Out[11]: $\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

Out[12]: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}$

- square diagonal matrices have interesting properties
  - multiplying a matrix A from the *left* with a diagonal matrix multiplies each *row* of A with the corresponding diagonal entry
  - multiplying a matrix A from the *right* with a diagonal matrix multiplies each *column* of A with the corresponding diagonal entry

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 8 & 10 & 12 \\ 21 & 24 & 27 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 9 \\ 4 & 10 & 18 \\ 7 & 16 & 27 \end{bmatrix}$$

identity matrix

- a special case is $\mathbf{I}$, the diagonal matrix with only 1s at the diagonal. It is called the **identity matrix**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

(NB: This is an example where commutativity happens to hold.)

- Strictly speaking, there is an $n \times n$ identity matrix for each number n of dimensions. In mathematical contexts, we usually rely that the value of n determined by the context. When programming, you have to be pedantic about theses things, of course.

inverse matrix

- Given a square matrix A, the **inverse Matrix** A^{-1} – if it exists – reduces A to $\mathbf{I}$.

$$AA^{-1} = A^{-1}A = \mathbf{I}$$

- Note that A^{-1} is both the left and the right multiplicative inverse.
- example:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

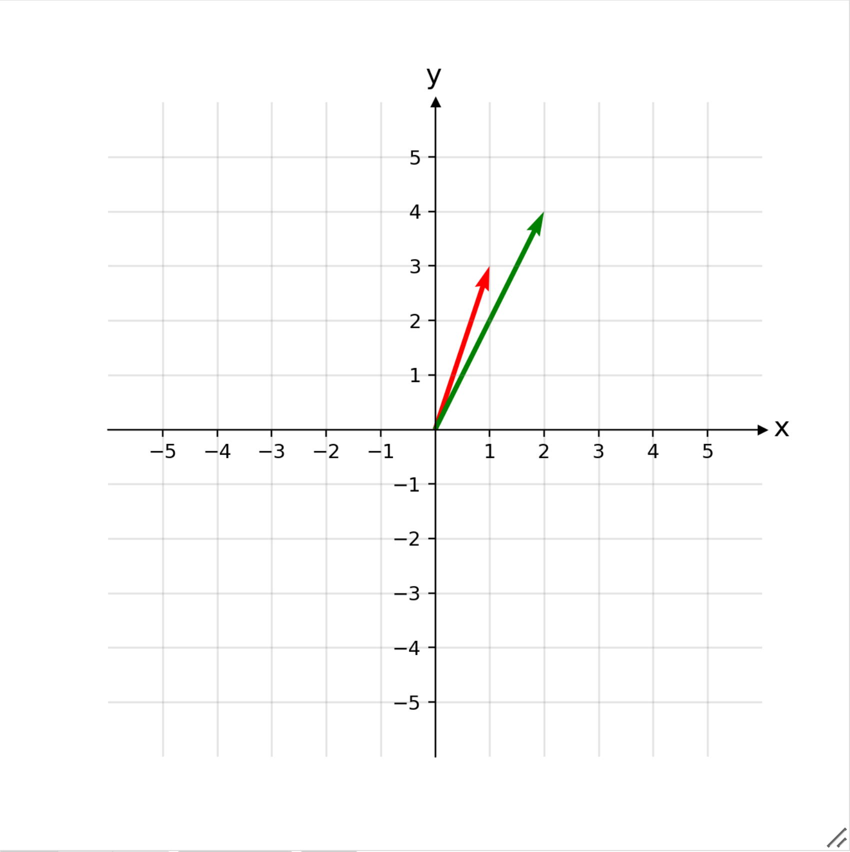
- example for a matrix without inverse:

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$
$$B^{-1} \text{ is undefined}$$

Why is this so?

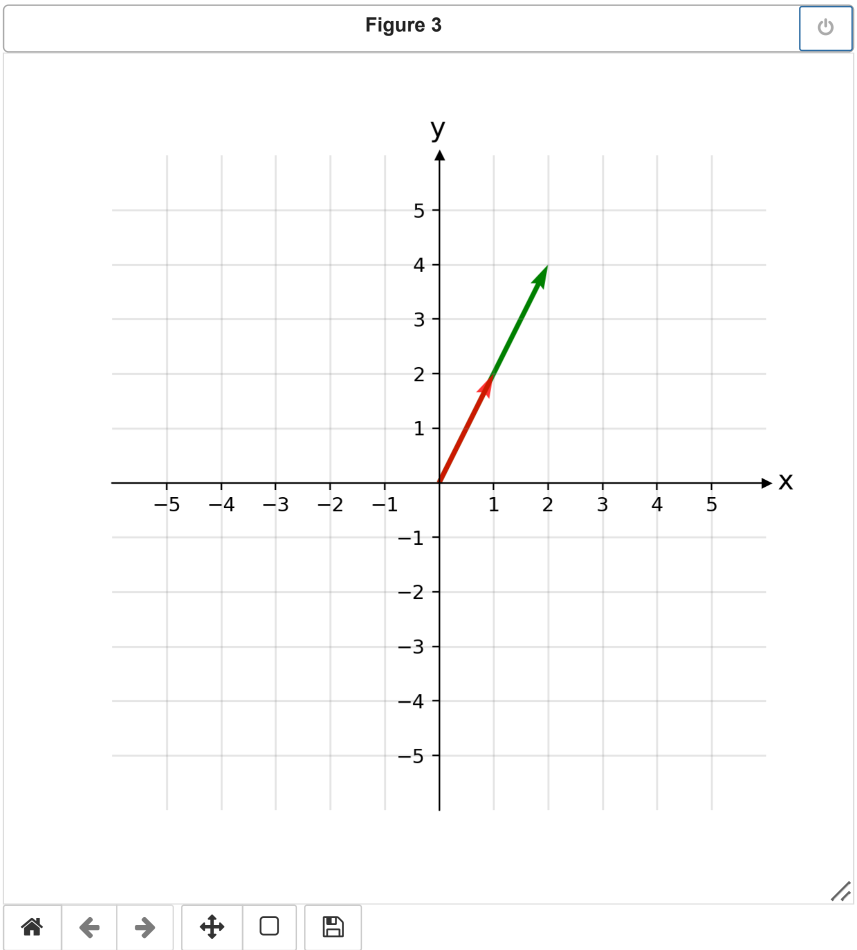
Example of the “good” (invertable) matrix:

Figure 2




```
Out[13]: <matplotlib.quiver.Quiver at 0x7fdddf6e76a0>
```

Example of the “bad” (non-invertable) matrix:



Out[14]: <matplotlib.quiver.Quiver at 0x7fdddf60f430>

examples of an invertable and a non-invertable matrix in 3d

$$A = \begin{bmatrix} 1 & -4 & 2 \\ -2 & 1 & 3 \\ 2 & 6 & 8 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} \frac{5}{63} & -\frac{22}{63} & \frac{1}{9} \\ -\frac{11}{63} & -\frac{2}{63} & \frac{1}{18} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{18} \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & -4 & 2 \\ -2 & 1 & 3 \\ 2 & 6 & -10 \end{bmatrix}$$

is undefined

Figure 4

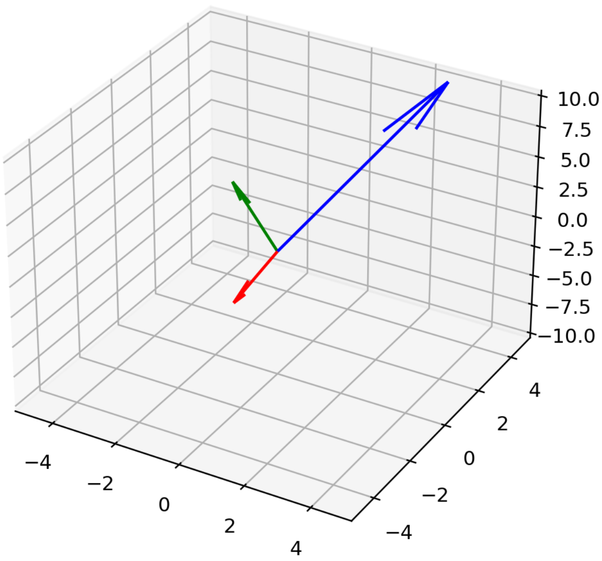
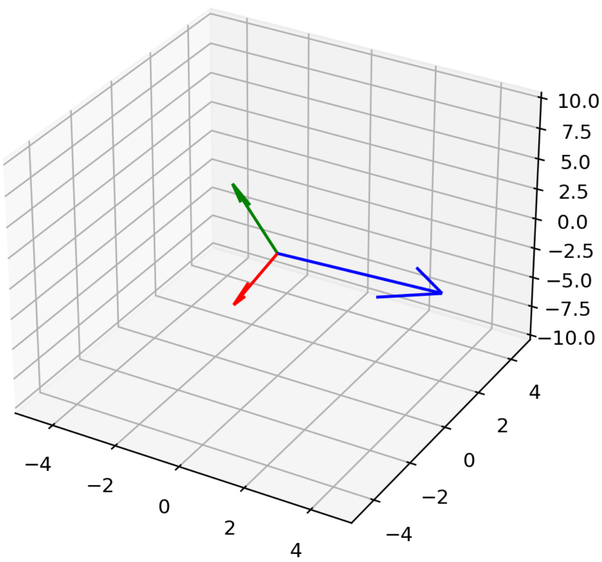


Figure 5



- a matrix is invertible if the **column space** has dimensionality n

- it is not invertible if the column space has dimensionality $< n$

Matrices with python and numpy

- vector

In [17]:
```
import numpy as np
x = np.array([1,2,3])
x
```

Out[17]:
```
array([1, 2, 3])
```

- matrix

In [18]:
```
A = np.array([
    [1,2, 3],
    [4,5,6],
    [7,8,9]
])
A
```

Out[18]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

vector algebra

In [19]:
```
y = np.array([3,10,4])

x + y
```

Out[19]:
```
array([ 4, 12,  7])
```

In [20]:
```
3*x
```

Out[20]:
```
array([3, 6, 9])
```

In [21]:
```
2*x - 3*y
```

Out[21]:
```
array([ -7, -26,  -6])
```

applying a matrix to a vector

In [22]:
```
A, x
```

Out[22]:
```
(array([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]),
 array([1, 2, 3]))
```

In [23]:
```
A @ x
```

Out[23]:
```
array([14, 32, 50])
```

matrix transposition

In [24]:
```
A
```

Out[24]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [25]:
```
A.T
```

Out[25]:
```
array([[1, 4, 7],
       [2, 5, 8],
       [3, 6, 9]])
```

matrix multiplication

In [26]:
```python
B = np.array([
    [4,1,0],
    [1,0,2],
    [4,5,6]
])
```

In [27]:
```python
A, B
```

Out[27]:
```
(array([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]),
 array([[4, 1, 0],
        [1, 0, 2],
        [4, 5, 6]]))
```

In [28]:
```python
A @ B
```

Out[28]:
```
array([[18, 16, 22],
       [45, 34, 46],
       [72, 52, 70]])
```

identity matrix

In [29]:
```python
np.eye(3)
```

Out[29]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

diagonal matrix

In [30]:
```python
np.diag([2,3, 4])
```

Out[30]:
```
array([[2, 0, 0],
       [0, 3, 0],
       [0, 0, 4]])
```

inverse matrix

In [31]:
```python
A = np.array(
[
    [1, -4, 2],
    [-2, 1, 3],
    [2,6,8]
])
A
```

Out[31]:
```
array([[ 1, -4,  2],
       [-2,  1,  3],
       [ 2,  6,  8]])
```

In [32]:
```python
np.linalg.inv(A)
```

Out[32]:
```
array([[ 0.07936508, -0.34920635,  0.11111111],
       [-0.17460317, -0.03174603,  0.05555556],
       [ 0.11111111,  0.11111111,  0.05555556]])
```

In [33]:
```python
B = np.array(
[
    [1, -4, 2],
    [-2, 1, 3],
    [2,6,-10]
])
```

In [34]:
```python
B
```

Out[34]:
```
array([[  1,  -4,   2],
       [ -2,   1,   3],
       [  2,   6, -10]])
```

In [35]:
```python
from numpy.linalg import LinAlgError

try:
    np.linalg.inv(B)
except LinAlgError:
    print("matrix is not invertible")
```

```
matrix is not invertible
```

Python and SymPy

In [36]:
```python
import sympy
from sympy import Matrix
```

- creating a vector

In [37]:
```python
x = Matrix([1,2,3])
x
```

Out[37]:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

- creating a matrix

In [38]:
```python
A = Matrix(
[
    [1,2, 3],
    [4,5,6],
    [7,8,9]
])
A
```

Out[38]:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

vector algebra

In [39]:
```python
y = Matrix([3,10,4])
y
```

Out[39]:

$$\begin{bmatrix} 3 \\ 10 \\ 4 \end{bmatrix}$$

In [40]:
```python
x+y
```

Out[40]:

$$\begin{bmatrix} 4 \\ 12 \\ 7 \end{bmatrix}$$

In [41]:
```python
3*x
```

Out[41]:

$$\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

In [42]:
```python
2 * x - 3 * y
```

Out[42]:

$$\begin{bmatrix} -7 \\ -26 \\ -6 \end{bmatrix}$$

applying a vector to a matrix

In [43]:
```
A * x
```

Out[43]:

$$\begin{bmatrix}14\\32\\50\end{bmatrix}$$

matrix transposition

In [44]:
```
A.T
```

Out[44]:

$$\begin{bmatrix}1 & 4 & 7\\2 & 5 & 8\\3 & 6 & 9\end{bmatrix}$$

matrix multiplication

In [45]:
```
B = Matrix(
[
     [4,1,0],
     [1,0,2],
     [4,5,6]
])
B
```

Out[45]:

$$\begin{bmatrix}4 & 1 & 0\\1 & 0 & 2\\4 & 5 & 6\end{bmatrix}$$

In [46]:
```
A * B
```

Out[46]:

$$\begin{bmatrix}18 & 16 & 22\\45 & 34 & 46\\72 & 52 & 70\end{bmatrix}$$

identity matrix

In [47]:
```
sympy.eye(3)
```

Out[47]:

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

diagonal matrix

In [48]:
```
sympy.diag(1,2,3)
```

Out[48]:

$$\begin{bmatrix}1 & 0 & 0\\0 & 2 & 0\\0 & 0 & 3\end{bmatrix}$$

matrix inverse

In [49]:
```
A = Matrix(
[
     [1, -4, 2],
     [-2, 1, 3],
     [2,6,8]
])
```

In [50]:
```
A.inv()
```
Out[50]:
$$\begin{bmatrix} \frac{5}{63} & -\frac{22}{63} & \frac{1}{9} \\ -\frac{11}{63} & -\frac{2}{63} & \frac{1}{18} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{18} \end{bmatrix}$$

In [51]:
```
B = Matrix(
[
    [1, -4, 2],
    [-2, 1, 3],
    [2,6,-10]
])
```

In [52]:
```
try:
    B.inv()
except ValueError:
    print("B is not invertible")
```

```
B is not invertible
```

Symbolic computation with SymPy

The real strength of Sympy is that it can calculate with variables as well as with numbers.

In [53]:
```
from sympy import symbols
a,b,c,d = symbols('a b c d')
```

In [54]:
```
A = Matrix([
    [a, b],
    [c,d]
])
A
```
Out[54]:
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

In [55]:
```
A.inv()
```
Out[55]:
$$\begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$