In [2]:
```
import sympy
from sympy import Matrix
import numpy as np
```

In [3]:
```
#%matplotlib notebook
%matplotlib widget
import matplotlib.pyplot as plt
```

# Mathematics for Machine Learning

## Session 01: Introduction


## Gerhard Jäger


## October 22, 2024

## Homework

Homework assignments have to be submitted as pdf files via Moodle. You can write them on the computer, but you can also write them by hand to upload fotos.

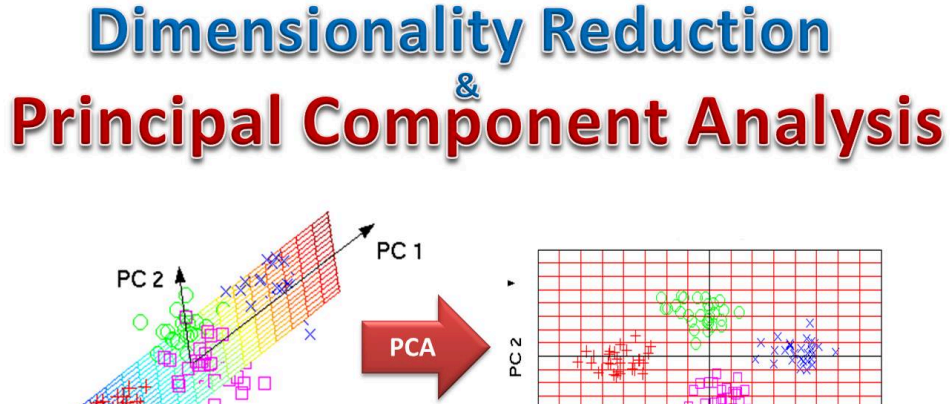
## Side remark

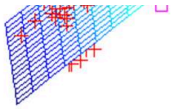
I will illustrate some concepts computationally, using *Python* and the packages *numpy* and *sympy*.

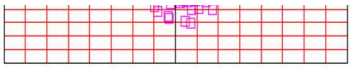
It is recommended for you to make yourself familiar with these packages.

# Applications of linear algebra

## Exploratory data analysis, e.g.

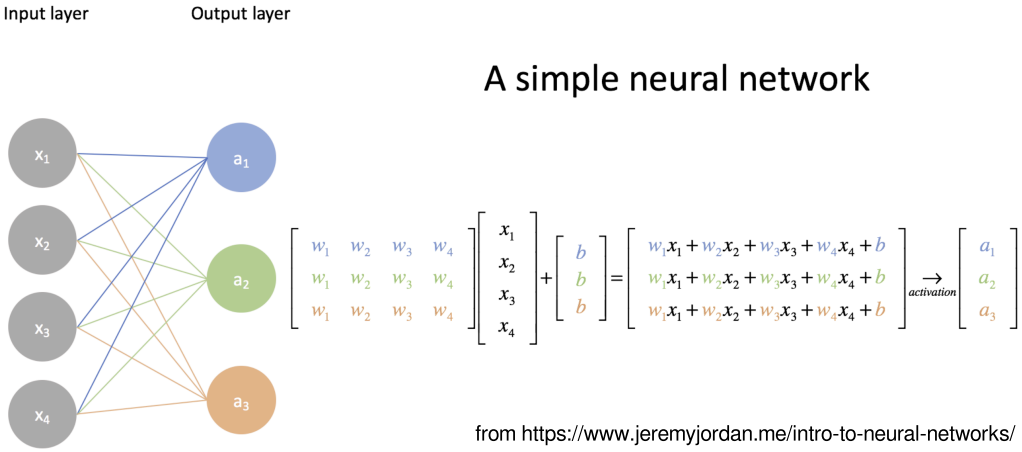







# Applications of linear algebra

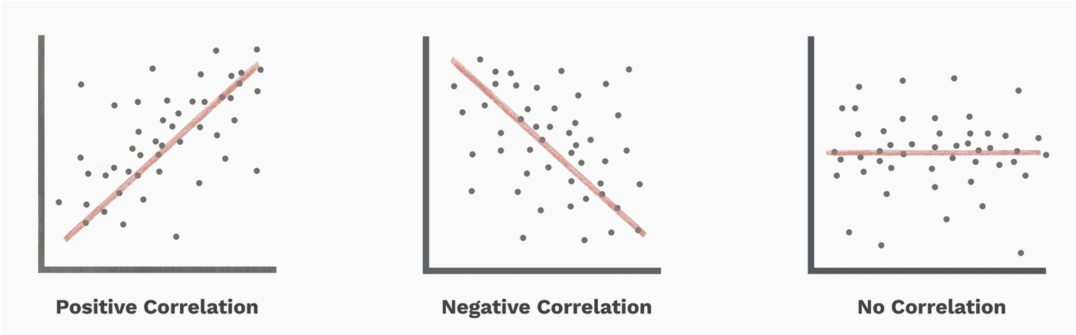
## Artificial Neural Networks




from https://www.jeremyjordan.me/intro-to-neural-networks/

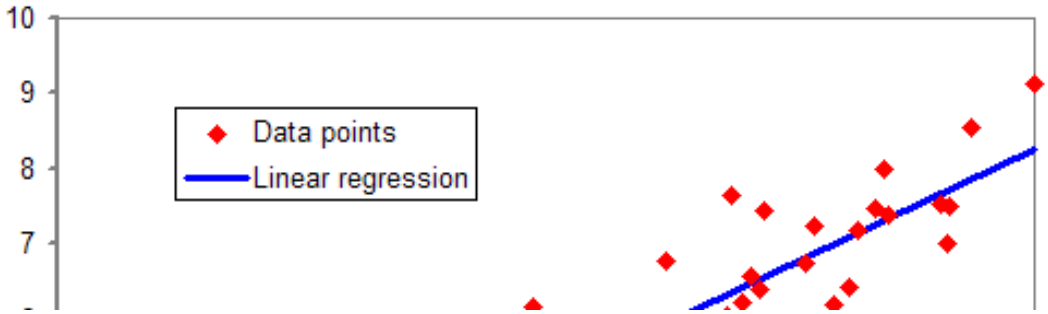
# Applications of linear algebra

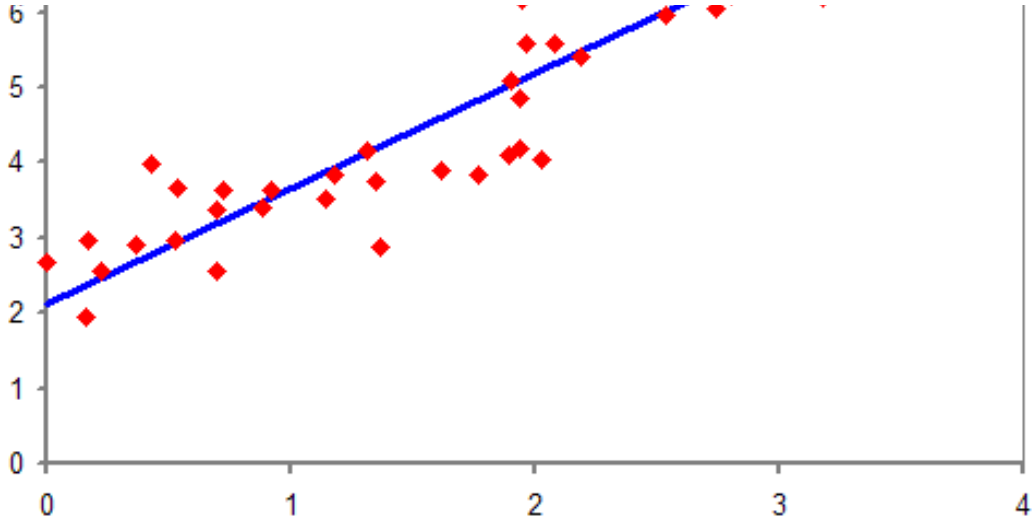
## Descriptive statistics, e.g correlation




# Applications of linear algebra

## Inferential statistics, e.g. linear regression

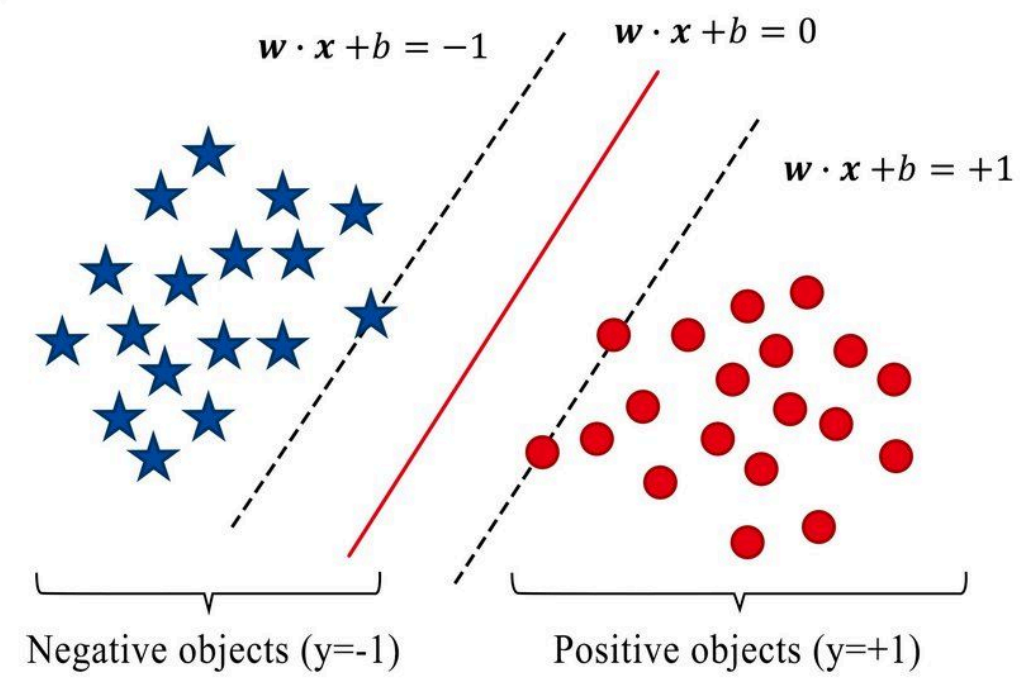





$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

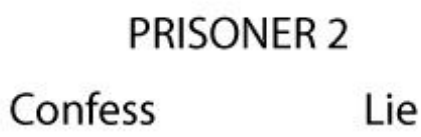
# Applications of linear algebra

## Machine learning




# Applications of linear algebra

## Game theory



| | | PRISONER 1 | |
|---|---|---|---|
| | Confess | -8 , -8 | 0 , -10 |
| | Lie | -10 , 0 | -1 , -1 |

$$E(u) = \tau' SRa$$

# Vectors and linear equations

- basis of linear algebra: solving **linear equations**
- Examples:

$$\begin{aligned} x + 2y + 3z &= 6 \\ 2x + 5y + 2z &= 4 \\ 6x - 3y + z &= 2 \end{aligned}$$

or

$$\begin{aligned} 2x - y &= 0 \\ -x + 2y &= 3 \end{aligned}$$

Let's focus on the second example. We can write this in **matrix notation**.

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

Schematically:

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

The goal is to find $x$ and $y$, (in general: $\mathbf{b}$) which solve the equation.

Two fundamental ways to approach this problem geometrically:

- **row picture**
- **column picture**

# Row picture

$$2x - y = 0$$

describes a line in a Cartesian plane. So does

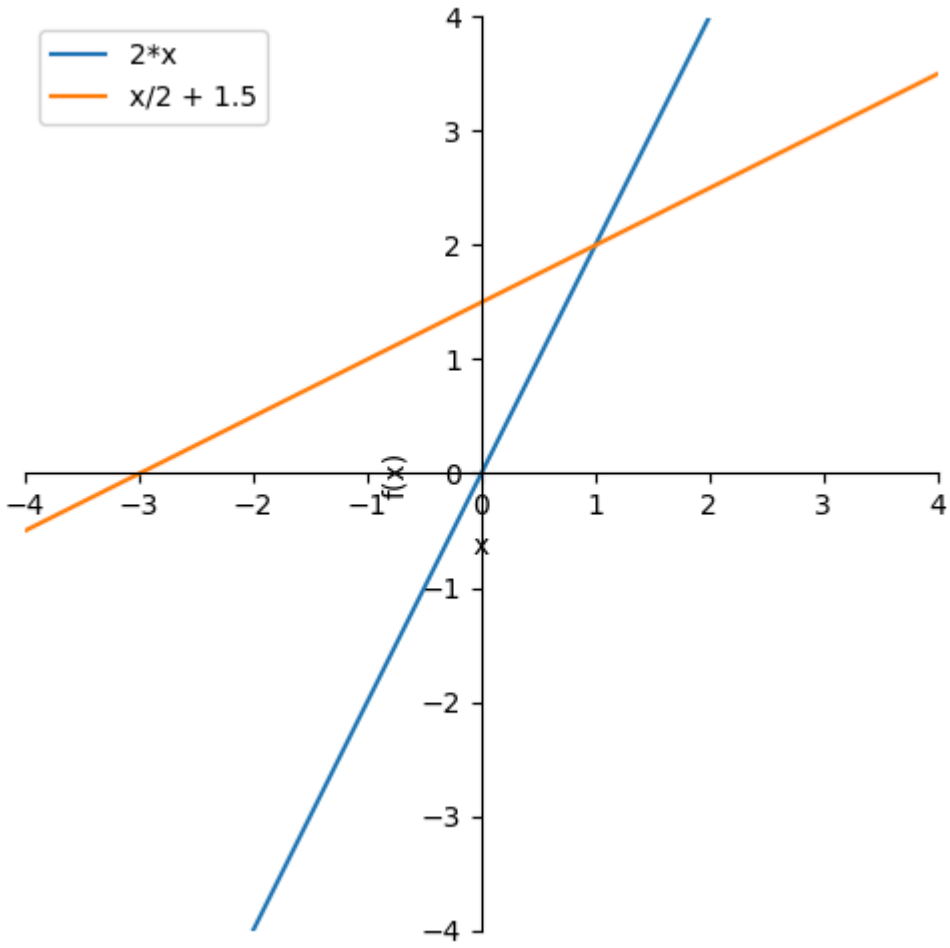
$$-x + 2y = 3$$

The solution is where the lines intersect.

In [4]:
```
from sympy import symbols
from sympy.plotting import plot
x = symbols('x')
y = symbols('y')
```

In [5]:
```
p = plot(2*x, x/2+3/2, legend=True, xlim=(-4,4), ylim=(-4,4), size=(5,5
```

Figure




# Column picture

recall

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

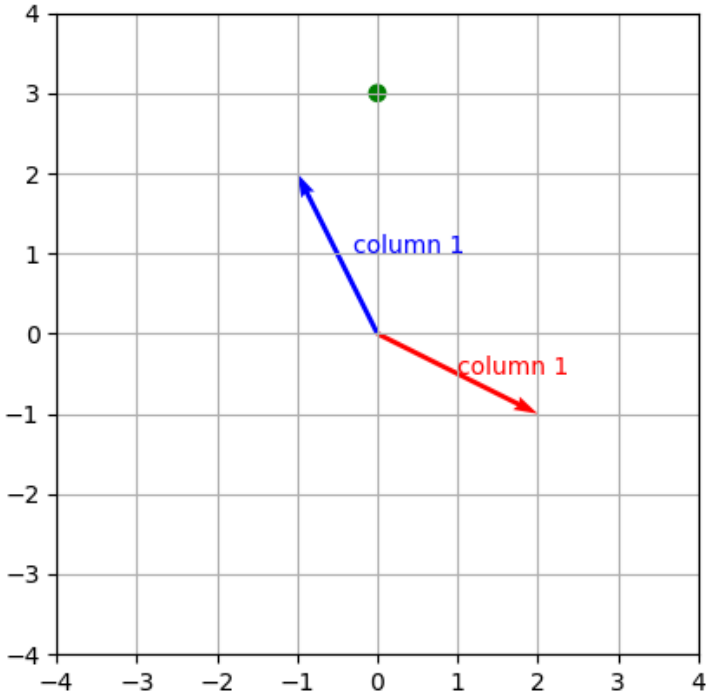
This can be decomposed into

$$\begin{pmatrix} 2 \\ \end{pmatrix} x + \begin{pmatrix} -1 \\ \end{pmatrix} y = \begin{pmatrix} 0 \\ \end{pmatrix}$$

\−1/ \ 2 / \3/

In [6]:
```
fig, ax = plt.subplots()
q1 = ax.quiver(0,0,2,-1, units='xy', scale=1, color='red')
q1 = ax.quiver(0,0,-1,2, units='xy', scale=1, color='blue')
c = plt.Circle((0,3), radius=.1, color='green')
ax.add_patch(c)
plt.grid()
ax.set_aspect('equal')
plt.xlim(-4,4)
plt.ylim(-4,4)
plt.annotate("column 1", (1,-.5), color='red')
plt.annotate("column 1", (-.3,1), color='blue')
plt.show()
```

Figure

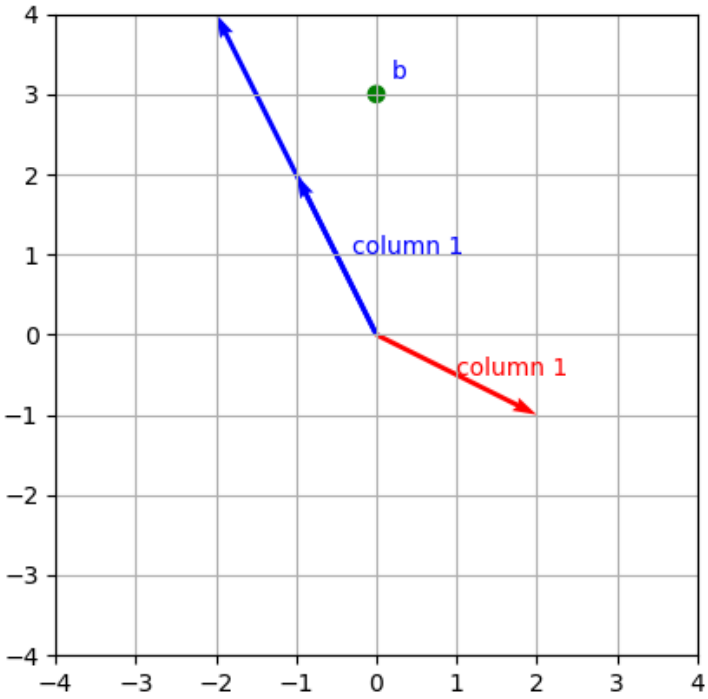



Vectors can be multiplied with real numbers.

Let us multiply the second column with 2.

In [7]:
```
fig, ax = plt.subplots()
q1 = ax.quiver(0,0,2,-1, units='xy', scale=1, color='red')
q2 = ax.quiver(0,0,-2,4, units='xy', scale=1, color='blue')
q3 = ax.quiver(0,0,-1,2, units='xy', scale=1, color='blue')
c = plt.Circle((0,3), radius=.1, color='green')
ax.add_patch(c)
plt.grid()
ax.set_aspect('equal')
plt.xlim(-4,4)
plt.ylim(-4,4)
plt.annotate("column 1", (1,-.5), color='red')
plt.annotate("column 1", (-.3,1), color='blue')
plt.annotate("b", (.2,3.2), color='blue')
plt.show()
```

Figure

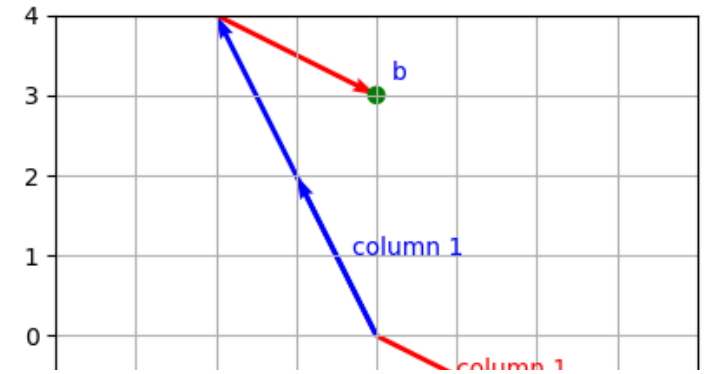


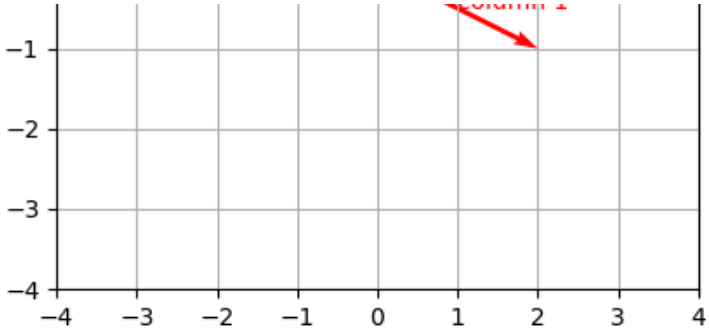

Vectors can also be added. Let's add column one to 2*(column 1).

In [8]:

```
fig, ax = plt.subplots()
q1 = ax.quiver(0,0,2,-1, units='xy', scale=1, color='red')
q2 = ax.quiver(0,0,-2,4, units='xy', scale=1, color='blue')
q3 = ax.quiver(0,0,-1,2, units='xy', scale=1, color='blue')
q4 = ax.quiver(-2,4,2,-1, units='xy', scale=1, color='red')
c = plt.Circle((0,3), radius=.1, color='green')
ax.add_patch(c)
plt.grid()
ax.set_aspect('equal')
plt.xlim(-4,4)
plt.ylim(-4,4)
plt.annotate("column 1", (1,-.5), color='red')
plt.annotate("column 1", (-.3,1), color='blue')
plt.annotate("b", (.2,3.2), color='blue')
plt.show()
```

Figure






- so our solution is

$$x = 1$$
$$y = 2$$

## 3 equations with three unknowns

$$x + 2y + 3z = 6$$
$$2x + 5y + 2z = 4$$
$$6x - 3y + z = 2$$

## row picture

A linear equation in three unknowns describes a **plane** in a 3-dimensional space.

In [9]:
```
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
```

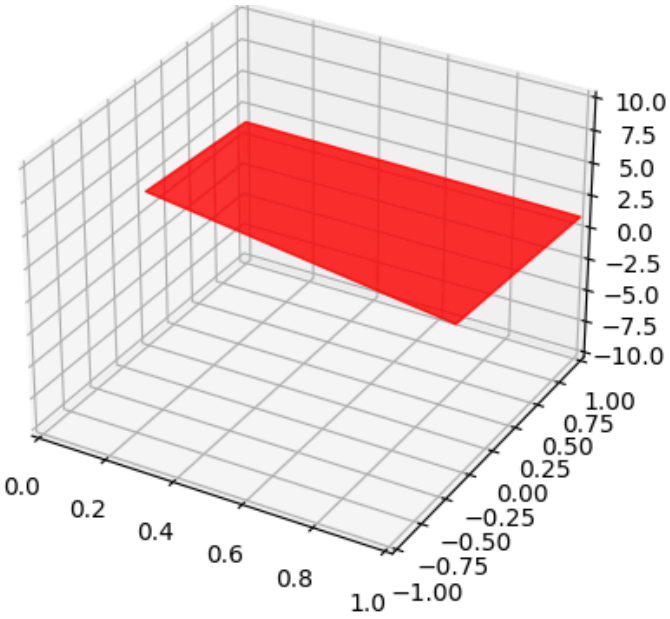
first row:

In [10]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = [0, 0,1,1]
y = [0, 1, 1, -4/11]
z1 = [2.0, 1.3333333333333335, 1.0, 1.9090909090909092]

vertices = [list(zip(x,y,z1))]
poly1 = Poly3DCollection(vertices, alpha=0.8, color='red')
ax.add_collection3d(poly1)
ax.set_xlim(0,1)
ax.set_ylim(-1,1)
ax.set_zlim(-10,10)
```

Out[10]:
```
(-10.0, 10.0)
```

Figure




second row

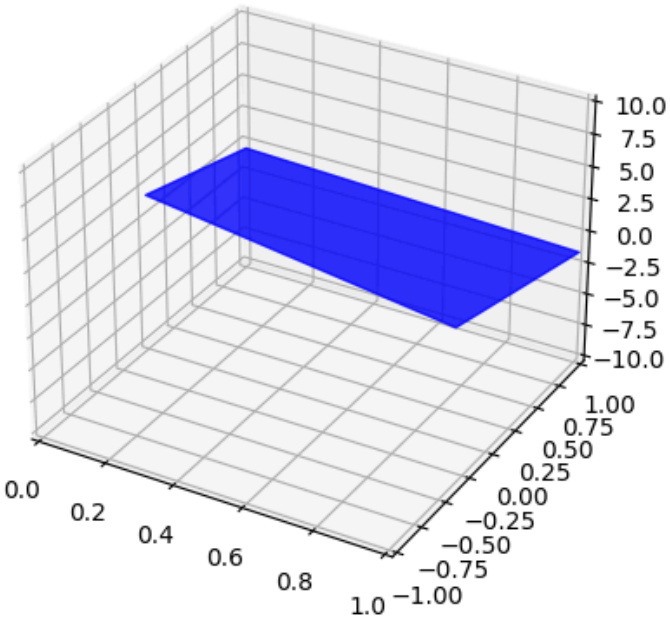
In [11]:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

z2 = [2.0, -0.5, -1.5, 1.9090909090909092]
vertices = [list(zip(x,y,z2))]
poly2 = Poly3DCollection(vertices, alpha=0.8, color='blue')
ax.add_collection3d(poly2)
ax.set_xlim(0,1)
ax.set_ylim(-1,1)
ax.set_zlim(-10,10)
```

Out[11]: (-10.0, 10.0)

Figure

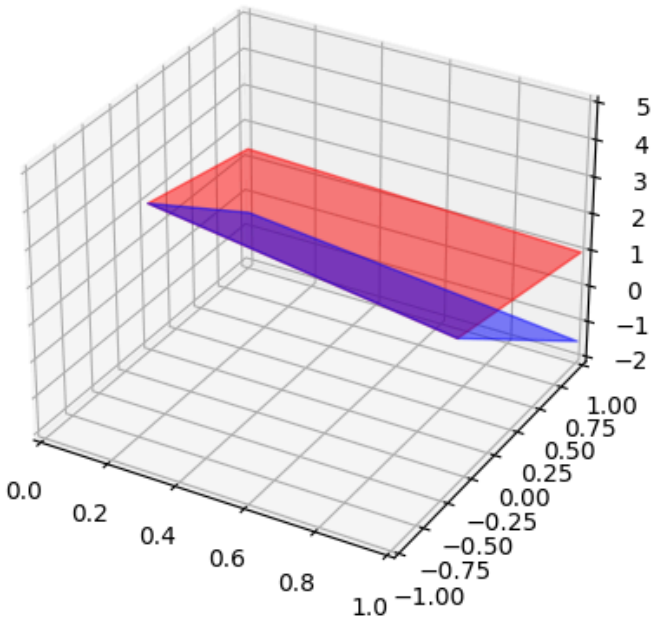


both rows together

In [12]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
vertices1 = [list(zip(x,y,z1))]
vertices2 = [list(zip(x,y,z2))]
poly1 = Poly3DCollection(vertices1, alpha=0.5, color='red')
poly2 = Poly3DCollection(vertices2, alpha=0.5, color='blue')
ax.add_collection3d(poly1)
ax.add_collection3d(poly2)
ax.set_xlim(0,1)
ax.set_ylim(-1,1)
ax.set_zlim(-2,5)
```

Out[12]:
```
(-2.0, 5.0)
```

Figure

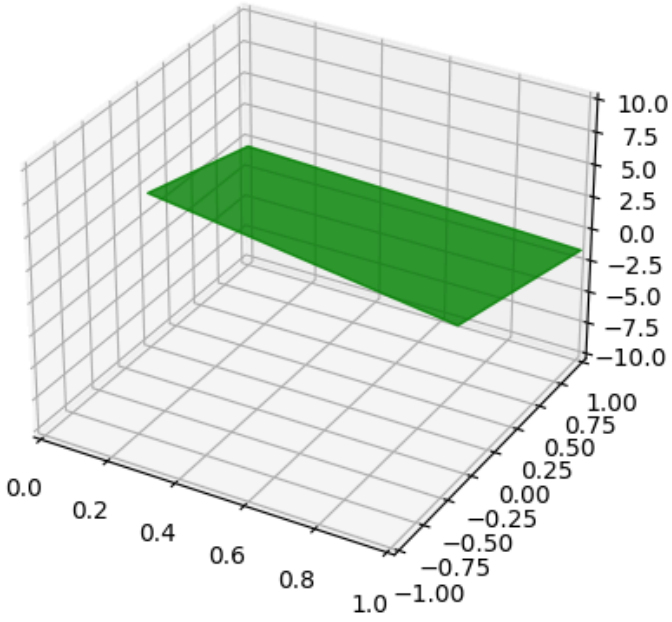



third row

In [13]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
z3 = [2, 5, -1, -5.090909090909091]
vertices3 = [list(zip(x,y,z3))]
poly3 = Poly3DCollection(vertices, alpha=0.8, color='green')
ax.add_collection3d(poly3)
ax.set_xlim(0,1)
ax.set_ylim(-1,1)
ax.set_zlim(-10,10)
```

Out[13]:
```
(-10.0, 10.0)
```

Figure




all three rows together

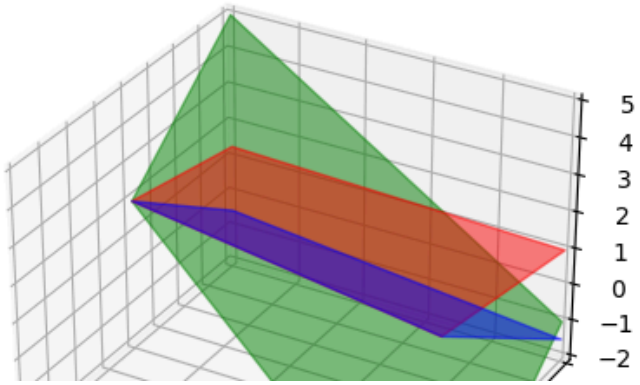
In [14]:

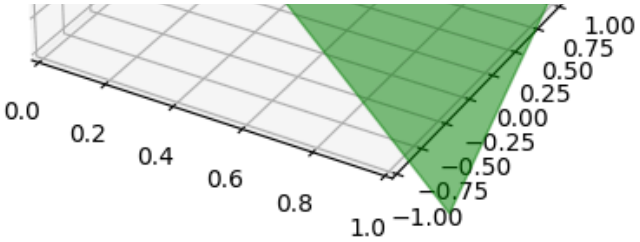
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
vertices1 = [list(zip(x,y,z1))]
vertices2 = [list(zip(x,y,z2))]
poly1 = Poly3DCollection(vertices1, alpha=0.5, color='red')
poly2 = Poly3DCollection(vertices2, alpha=0.5, color='blue')
poly3 = Poly3DCollection(vertices3, alpha=0.5, color='green')
ax.add_collection3d(poly1)
ax.add_collection3d(poly2)
ax.add_collection3d(poly3)
ax.set_xlim(0,1)
ax.set_ylim(-1,1)
ax.set_zlim(-2,5)
```

Out[14]:

```
(-2.0, 5.0)
```

Figure






- the solution set for each row is a **plane**
- the intersection of two plane (if they are not parallel) is a **line**
- the intersection of three planes is a **point**

This point is the solution of the system of equations.

## column picture

recall the system to be solved:

$$\begin{aligned} x + 2y + 3z &= 6 \\ 2x + 5y + 2z &= 4 \\ 6x - 3y + z &= 2 \end{aligned}$$

- can be rewritten as

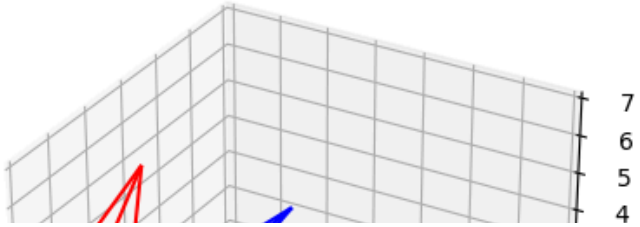
$$\begin{pmatrix} 1 \\ 2 \\ 6 \end{pmatrix} x + \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix} y + \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} z = \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix}$$

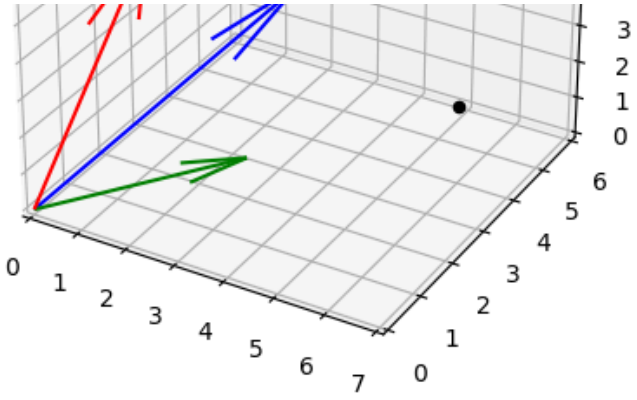
In [15]:

```
import mpl_toolkits.mplot3d.art3d as art3d

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.quiver((0,), (0,), (0,), (1,), (2,), (6,), color='red')
ax.quiver((0,), (0,), (0,), (2,), (5,), (3,), color='blue')
ax.quiver((0,), (0,), (0,), (3,), (2,), (1,), color='green')
ax.scatter((6,), (4,), (2,), color='black')
ax.set_xlim([0, 7])
ax.set_ylim([0, 6])
ax.set_zlim([0,7])
plt.show()
```

Figure






- We can see with the bare eye that **b** is a multiple of the third column vector. So the solutions for $x$ and $y$ are $0$. The solution for $z$ happens to be 2.

We have

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

$$\mathbf{b} = \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix}$$

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

The solution is

$$\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

## Big question?

- Is there a solution of this equation for every $\mathbf{b}$?

- What would $A$ have to look like to get a different answer?

# Back to the basics

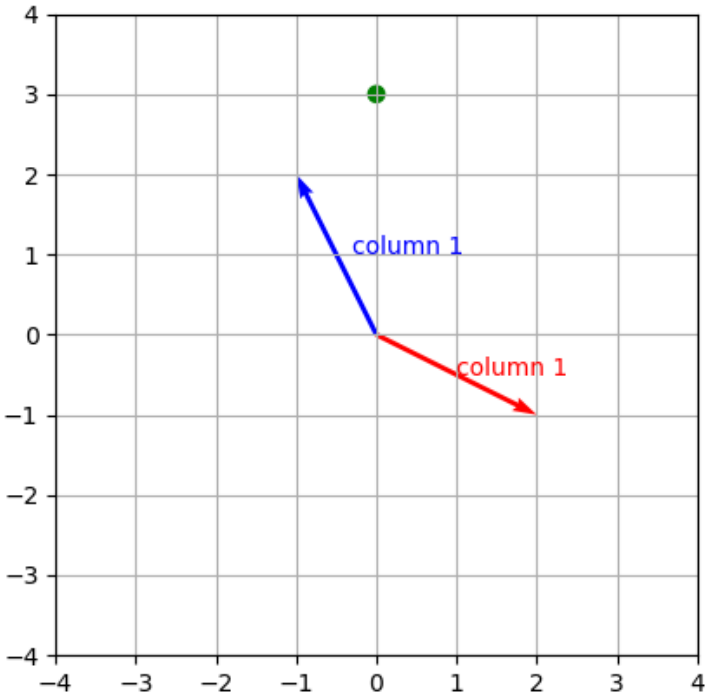
- a **vector** is an ordered sequence of $n$ real numbers
- geometrically, a vector can be interpreted as a point in the $n$-dimensional space
- 2 dimensions equals plane

In [16]:

```
fig, ax = plt.subplots()
q1 = ax.quiver(0,0,2,-1, units='xy', scale=1, color='red')
```

```
q1 = ax.quiver(0,0,-1,2, units='xy', scale=1, color='blue')
c = plt.Circle((0,3), radius=.1, color='green')
ax.add_patch(c)
plt.grid()
ax.set_aspect('equal')
plt.xlim(-4,4)
plt.ylim(-4,4)
plt.annotate("column 1", (1,-.5), color='red')
plt.annotate("column 1", (-.3,1), color='blue')
plt.show()
```

Figure



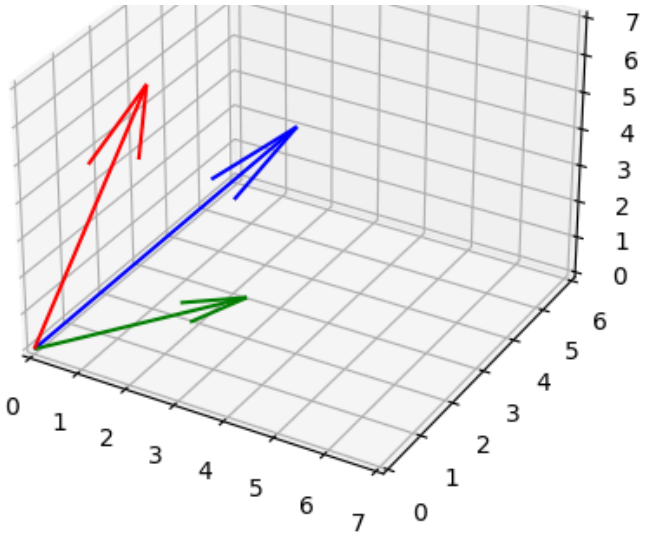

- 3 dimension equals space

In [17]:

```
import mpl_toolkits.mplot3d.art3d as art3d

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.quiver((0,), (0,), (0,), (1,), (2,), (6,), color='red')
ax.quiver((0,), (0,), (0,), (2,), (5,), (3,), color='blue')
ax.quiver((0,), (0,), (0,), (3,), (2,), (1,), color='green')
ax.set_xlim([0, 7])
ax.set_ylim([0, 6])
ax.set_zlim([0,7])
plt.show()
```

Figure




- higher dimensions are hard to visualize, but beyond that, there is nothing special about them
- vectors are usually written as bold-faced lowercase letter, like $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}, \mathbf{v}$
- when the individual cells are spelled out, a vector is written als a *column*

- individual components of a vector are indicated by subscript. If

$$\mathbf{x} = \begin{pmatrix} 8 \\ 6 \\ 1 \end{pmatrix},$$

then

$$\begin{aligned} x_1 &= 8 \\ x_2 &= 6 \\ x_3 &= 1 \end{aligned}$$

## vector operations

- vectors can be **added**, provided they have the same length/dimensionality

$$\mathbf{x} + \mathbf{y} \doteq \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

- vector addition is
  - commutative

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

- associative

$$(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$$

## vector operations

- vectors can be **multiplied with real numbers**

$$a \cdot \mathbf{x} \doteq \begin{pmatrix} a \cdot x_1 \\ a \cdot x_2 \\ \vdots \\ a \cdot x_n \end{pmatrix}$$

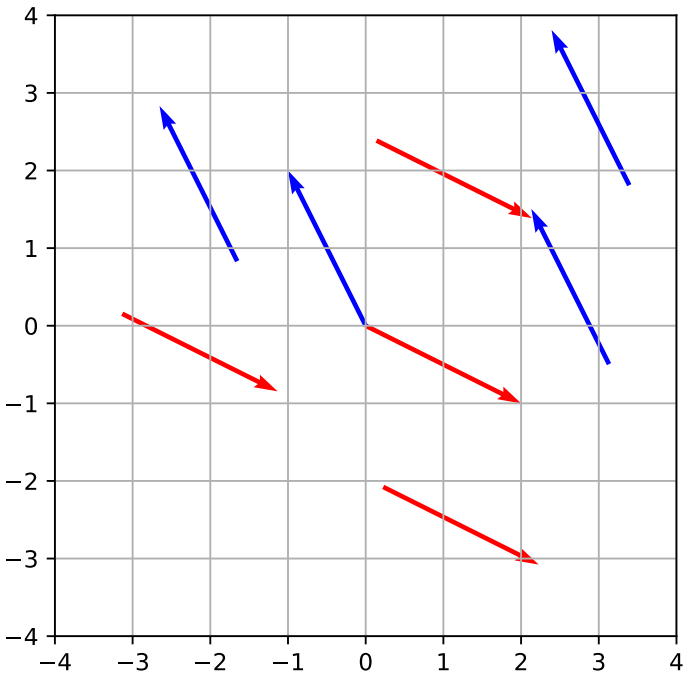
Real numbers are often called **scalars**, to distinguish them from vectors. Multiplication of a vector with a scalar is called **scalar multiplication**.

Scalar multiplication and vector addition obey the **distributive law**:

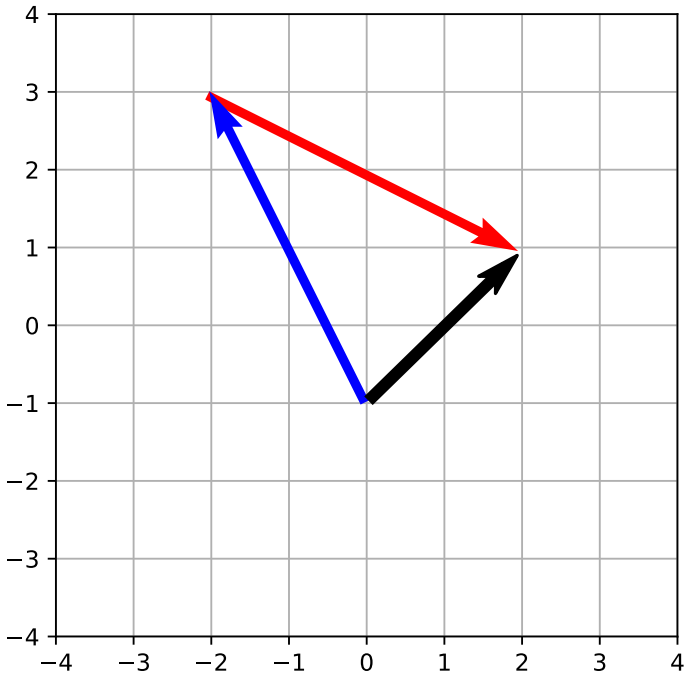
$$a \cdot (\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$$

## geometric interpretation

- vectors can also be seen as *equivalence classes of pairs of points* in the $n$-dimensional space
- often drawn as an arrow
- two arrows represent the same vector if they have the same length and direction



- vector addition $\mathbf{x} + \mathbf{y}$ amounts to moving the start point of $\mathbf{y}$ to the end point of $\mathbf{x}$ and connecting the start point of $\mathbf{x}$ to the end point of $\mathbf{y}$.




## important questions:

- Let $\mathbf{x} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. What is the set of vectors $\{a\mathbf{x} | a \in \mathbb{R}\}$?
- Let $\mathbf{y} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. What is the set of vectors $\{a\mathbf{x} + b\mathbf{y} | a, b \in \mathbb{R}\}$?

A **linear combination** of vectors is the result of applying scalar multiplication and vector addition to them.

So the last question amounts to: *What is the set of linear combinations of* $\mathbf{x}$ *and* $\mathbf{y}$*?*

## inner product

- 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 proudct 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:

In [18]:
```
Matrix([0.6, 0.8])
```

Out[18]:
$$\begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}$$

In [19]:
```
from sympy import Rational
Matrix([
    Rational(1,3),
    Rational(2,3),
    Rational(2,3)
])
```

Out[19]:
$$\begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ \frac{2}{3} \end{bmatrix}$$

In [20]:
```
Matrix([
    sympy.Rational(1,2),
    sympy.Rational(1,2),
    sympy.Rational(1,2),
```

```
    sympy.Rational(1,2)
])
```

Out[20]:

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

In [21]:

```
x, y, z, t = sympy.symbols('x y z t')
Matrix([sympy.sin(x), sympy.cos(x)])
```

Out[21]:

$$\begin{bmatrix} \sin(x) \\ \cos(x) \end{bmatrix}$$

In [22]:

```
Matrix([0, 0, 0, 1, 0])
```

Out[22]:

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

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

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