STEM Day 2025, University of Dundee

Overview

The exercises below allow us to explore and use a number of important areas of mathematics:

- algebra
- geometry
- programming (I have purposefully left Python codes in the document)

Key skills are use of:

- examples to develop intuition
- proof to make general statements
- computer programming to help solve and visualise

Functions of one variable

You might have come across functions of one variable. For example, suppose that

$$f(x) = x + 2, \quad x \in \mathfrak{R}.$$

This states that the function f maps a point x to the point $x + 2$.

In this session we will represent transformations using the notation

$$x' = x + 2,$$

i.e. an old point, x, gets mapped to a new point, x'.

In the above example, for a given input value of x, we add 2 to get the output, x' (see graph of $f(x)$ in Figure 1).

```
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(0,5,20) # define discrete set of values ranging from 0 to 5

x_p=x+2 # define the new value of x

# Plot old v new
fig,ax=plt.subplots()
ax.plot(x,x)
ax.plot(x,x_p)
ax.set_xlabel('$x$')
ax.set_ylabel('$x\'$')
plt.show()
```

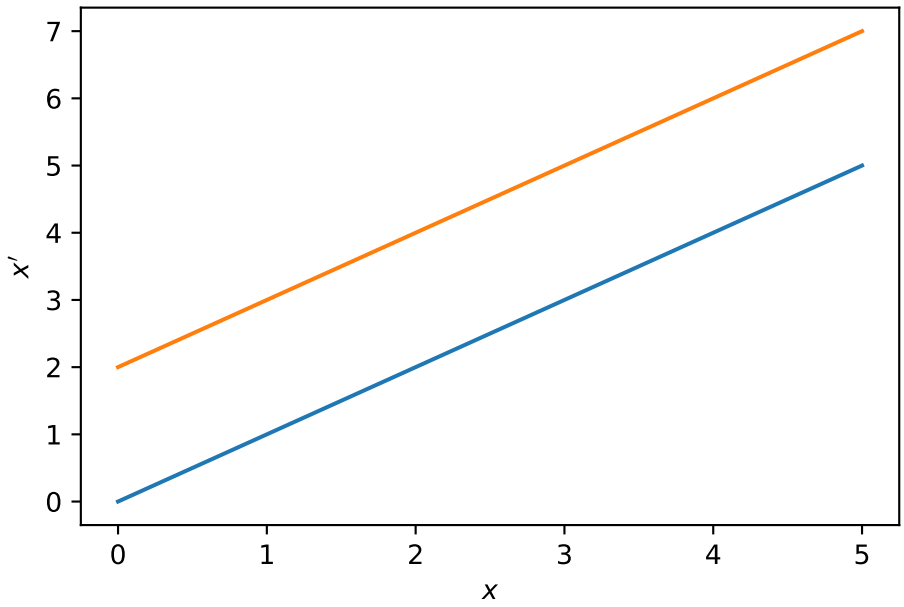


Figure 1

Mapping coordinates in the plane

Now consider a point (x, y) that sits in the 2D Cartesian plane. We want to consider transformations that map the coordinates of the point (x, y) to some other point in the plane (x', y').

Translations

Consider the map:

$$x' = x + 2$$
$$y' = y + 3.$$

This transformation states that we add 2 to each x value and 3 to each y value.

1. Calculate the transformation of the point (1,1). Confirm that this is consistent with Figure 2.
2. Compute where the point (2,2) gets mapped to and sketch this on Figure 2.
3. Now try (3,3). Can you spot a pattern in the mapped points? Based on your observation, are you willing to formulate a general statement about translations?

```
# define a set of old point
old_coordinates=[[1,1]]
old_coordinates=np.array(old_coordinates)
new_coordinates=old_coordinates.copy()

# make figure and plot old coordinates
fig,ax=plt.subplots()
for i in range(1):
    ax.plot(old_coordinates[i,0],old_coordinates[i,1],'k.',alpha=0.15+old_coordinates[i,1]/6
ax.set_xlim([0,14])
ax.set_ylim([0,14])
ax.grid()
ax.set_xticks(np.linspace(0,12,13))
ax.set_yticks(np.linspace(0,12,13))

# Compute transformed points
for i in range(old_coordinates.shape[0]):

    homog_vec=np.array(old_coordinates[i,:]).copy()
    new_coordinates[i,:]=homog_vec #+np.array([2,3])
    new_coordinates[i,1]=new_coordinates[i,1]+3
    new_coordinates[i,0]=new_coordinates[i,0]+2

# PLot transformed points
for i in range(1):
    ax.plot(new_coordinates[i,0],new_coordinates[i,1],'b.',markersize=24 )
```

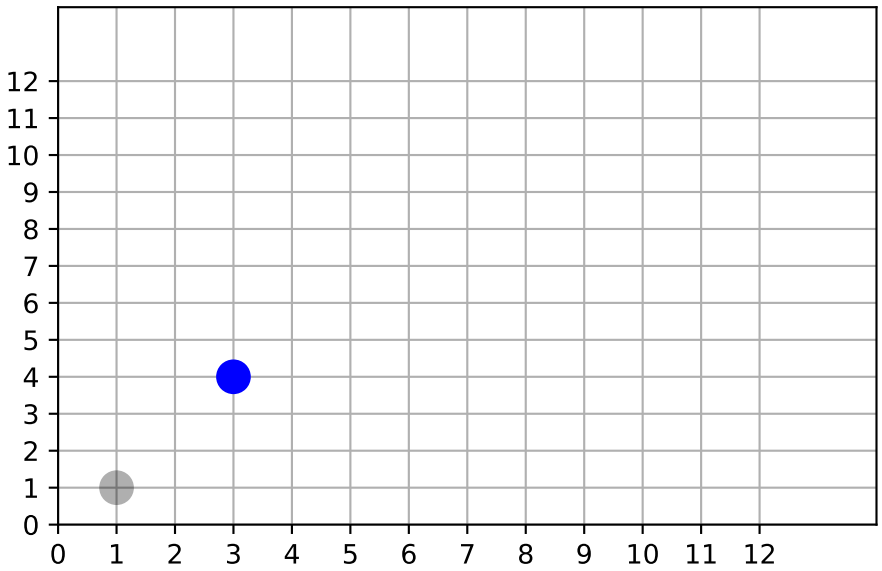


Figure 2

Translations map straight lines to straight lines of equal slope

Consider an infinite set of points that sit on a line

$$y = mx + c$$

where m and c are real constants. We want to show that, in general, translations map straight lines onto straight lines.
To do this, let's substitute in the parametric form of the straight line into the definition of the transformation, i.e.

$$y' = y + 3 = mx + c + 3.$$

We also have that

$$x' = x + 2 \implies x = x' - 2.$$

Hence

$$y' = mx + c + 3 = m(x' - 2) + c + 3 = mx' + (c + 3 - 2).$$

Hence the translation maps a straight line with slope m onto another straight line that also has slope m but a modified intercept.
Note that we have moved from an example to a general statement about all straight lines in $\mathfrak{R}^2$.

Could you show that this is true for a general translation

$$x' = x + a$$
$$y' = y + b$$

where a and b are real constants.

> **Note**
>
> It is common in mathematics to experiment with examples and then to prove results for more general cases.

Scalings

Let's consider a different type of transformation. The idea now is to take each of the points in the above example and multiply them by a scalar, d.

We can represent this type of transformation as

$$x' = dx,$$
$$y' = dy$$

Let's apply the same steps as in the translation case to this transformation:

1. Sketch where individual points get mapped to for a scaling where $d = 3$ (you could do this on Figure 2 again).
2. Use the sketch to show that a line of points maps on to another line of points under scaling.
3. In general, do scaling transformation preserve the slope?

Rotations

Now let's consider an anti-clockwise rotation about the origin.

This can be represented by

$$x' = \cos\theta x - \sin\theta y$$
$$y' = \sin\theta x + \cos\theta y$$

1. Consider a rotation given by $\theta = \pi/4$ (i.e. 45 degrees). Substitute for θ in the transformation and calculate and sketch where a point (1,1) transforms to. Note $\cos\pi/4 = \sin\pi/4 = 1/\sqrt{2}$.

2. Where do the points (2,2) and (3,3) map to under this rotation? Can you sketch these in the plane? Do rotations map straight lines onto straight lines.
3. Now consider a rotation of $\pi/2$ (i.e. 90 degrees).
4. Can you show that for a general rotation θ, a straight line of points maps onto a different straight line of points?

General linear transformations in the plane

With a little more work it can be shown that a whole family of transformations (rotations, translations, shear, scaling) can be represented by a single matrix multiplication. This technique is a powerful tool used in computer graphics.

1. Follow the link in this QR code

2. Go to the *Affine transformations in the plane* app.
3. Explore combining the different linear transformations described above. Is the movement of the square under different transformations consistent with your observations above?

Transforming images

A 2D grayscale image is represented by matrix of numbers. The numerical values at each voxel represents the intensity.

So, for example, a (small) image with a central spot could be represented by a matrix:

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

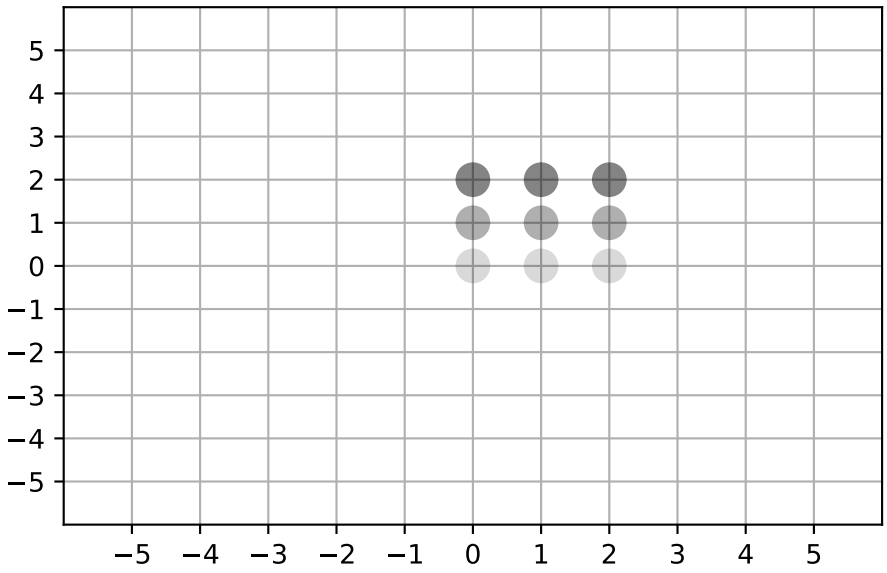
Let's consider a 3x3 grid with coordinates on an integer lattice (see below). The shading represents the intensity values in an image (i.e. there is vertical gradient in the image).

```
old_coordinates=[[0,0],[0,1],[0,2],[1,0],[1,1],
        [1,2],[2,0],[2,1],[2,2]]
old_coordinates=np.array(old_coordinates)
new_coordinates=old_coordinates.copy()

fig,ax=plt.subplots()
for i in range(9):
    ax.plot(old_coordinates[i,0],old_coordinates[i,1],'k.',alpha=0.15+old_coordinates[i,1]/6
ax.set_xlim([-6,6])
ax.set_ylim([-6,6])
ax.grid()
ax.set_xticks(np.linspace(-5,5,11))
ax.set_yticks(np.linspace(-5,5,11))

for i in range(old_coordinates.shape[0]):

    homog_vec=np.array(old_coordinates[i,:]).copy()
    new_coordinates[i,:]=homog_vec+np.array([2,3])
```



1. Let's apply a translation given by
$$x' = x + 2$$
$$y' = y + 3$$
to the image. To do this, transform each coordinate and 'carry' the shading. What properties does the transformed image have?

2. Now rotate the image anticlockwise about the origin by $\pi/4$ radians (i.e. 45 degrees). Describe what has happened to the image's orientation?

> **Note**
>
> We have learned that a simple image can be transformed using linear transformations. Each voxel of the image 'carries' the intensity value to a new location in the transformed image.

Nonlinear transformations

All the transformations considered above have been linear (x' and y' are some linear functions of x and y). It can be shown that this linearity results in certain properties always holding in the transformed image (e.g. straight lines map onto straight lines).

We can also consider nonlinear image mappings. Suppose that

$$x' = x + 3$$
$$y' = y\sqrt{x^2 + y^2}.$$

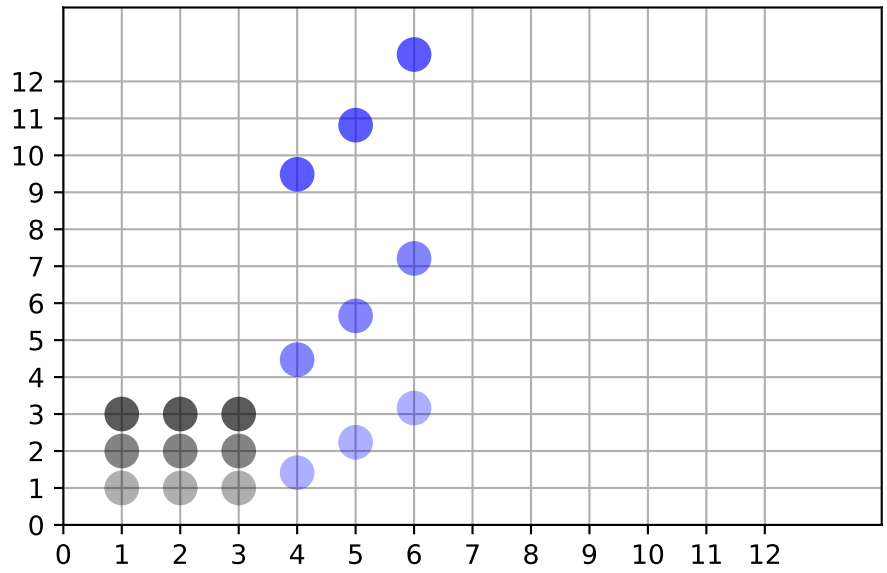
Here the mapping of the y coordinate depends on how close the original point is to the origin.

```
old_coordinates=[[1,1],[1,2],[1,3],[2,1],[2,2],
        [2,3],[3,1],[3,2],[3,3]]
old_coordinates=np.array(old_coordinates).astype(float)
new_coordinates=old_coordinates.copy()

fig,ax=plt.subplots()
for i in range(9):
    ax.plot(old_coordinates[i,0],old_coordinates[i,1],'k.',alpha=0.15+old_coordinates[i,1]/6
ax.set_xlim([0,14])
ax.set_ylim([0,14])
ax.grid()
ax.set_xticks(np.linspace(0,12,13))
ax.set_yticks(np.linspace(0,12,13))

for i in range(old_coordinates.shape[0]):

    homog_vec=np.array(old_coordinates[i,:]).copy()
    new_coordinates[i,:]=homog_vec #+np.array([2,3])
    new_coordinates[i,1]=new_coordinates[i,1]*np.sqrt(old_coordinates[i,1]**2+old_coordinates
    new_coordinates[i,0]=new_coordinates[i,0]+3
for i in range(9):
    ax.plot(new_coordinates[i,0],new_coordinates[i,1],'b.',alpha=0.15+old_coordinates[i,1]/6
```



The original grid (image) now gets distorted under the transformation. With more complicated nonlinear functions, we can begin to generate more complex distortions.

Use the QR code above to find another interactive demo called *D'Arcy Thompson and 2D mappings*