Euclidean Distances

Jeffrey Hewitt
CSCI 004

December 2, 2025

1 Introduction

In this paper I will be exploring the distances between points in different Euclidean spaces using Cartesian coordinate systems and their corresponding formulas. We will start in one dimension, then build up to three dimensions with a brief touch on complex values in a two dimensional space. Finally, these equations will be adapted into Python functions.

2 One Dimension

A one dimensional Euclidean system can be represented by a number line, x, ranging from negative infinity to positive infinity. The distance between two points, $A(x_1)$ and $B(x_2)$, is the absolute value of their difference as shown in Equations 1 and 2.

$$D_1(A, B) = D_1(x_1, x_2) = |x_1 - x_2| \tag{1}$$

$$D_1(2, 5) = |2 - 5| = |-3| = 3 \tag{2}$$

3 Two Dimensions

In two dimensions there are *two* number lines (axes), x and y, that intersect perpendicularly at the zero points, forming the "origin" at $(0, 0)$. To determine the distance between $C(x_1, y_1)$ and $D(x_2, y_2)$, we can take advantage of the Pythagorean Theorem. First we will need a third point $E(x_3, y_3)$ where $x_3 = x_1$ and $y_3 = y_2$, to form right triangle $\triangle CDE$.[1] The lengths of segments CE and DE can be measured using the one dimensional distance formula from Equation 1 on the x and y values respectively, seen in Equation 3. $\overline{CD}$ can then be found by plugging the side lengths into our Pythagorean formula as a and b and solving for c, demonstrated in Equation 4 and generalized in Equation 5.[2]

$$\begin{aligned} \overline{CE} &= |y_1 - y_3| = |y_1 - y_2| \\ \overline{DE} &= |x_2 - x_3| = |x_2 - x_1| \end{aligned} \tag{3}$$

$$\begin{aligned} c^2 &= a^2 + b^2 \\ (\overline{CD})^2 &= (\overline{CE})^2 + (\overline{DE})^2 \\ \overline{CD}^2 &= |y_1 - y_2|^2 + |x_2 - x_1|^2 \\ \overline{CD} &= \sqrt{|y_1 - y_2|^2 + |x_2 - x_1|^2} \end{aligned} \tag{4}$$

$$D_2(C, D) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5}$$

[1] $x_3 = x_2$ and $y_3 = y_1$ is equally valid, but is not used for this example. Likewise for three dimensions.

[2] Equation 5 is optimized using these facts: $a + b = b + a$, $|a - b| = |b - a|$, and $a^2 = (-a)^2$

4 Three Dimensions

Three dimensional space adds another axis, z, perpendicular to the x and y axes. For points $F(x_1, y_1, z_1)$ and $G(x_2, y_2, z_2)$, we can again create a third point, $H(x_3, y_3, z_3)$, letting $x_3 = x_1$, $y_3 = y_2$, and $z_3 = z_1$. Now we are able to calculate the side lengths of $\triangle FGH$. For $\overline{FH}$, the one dimensional distance formula, (D_1) from Equation 1 can be used. For $\overline{GH}$, the x and z values can be inserted into D_2 because the y-values are the same. Equation 6 shows both of these steps. With these sides evaluated, we are ready to use the Pythagorean Theorem in Equation 7 and construct a three dimensional distance formula in Equation 8.

$$\begin{aligned} \overline{FH} &= |y_1 - y_3| = |y_1 - y_2| \\ \overline{GH} &= \sqrt{(x_2 - x_3)^2 + (z_2 - z_3)^2} \\ &= \sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2} \end{aligned} \tag{6}$$

$$\begin{aligned} (\overline{FG})^2 &= (\overline{FH})^2 + (\overline{GH})^2 \\ \overline{FG}^2 &= |y_1 - y_2|^2 + \left(\sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2}\right)^2 \\ \overline{FG} &= \sqrt{|y_1 - y_2|^2 + (x_1 - x_2)^2 + (z_1 - z_2)^2} \end{aligned} \tag{7}$$

$$D_3(F, G) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{8}$$

5 Complex Values

A complex plane can be created using two axes, one representing all real numbers and the other containing all imaginary values. The coordinates have a *real* and a *complex* component in the form $p \pm qi$, where p and q are constants. Determining the distance between two points in this system is the same as with an "ordinary" two dimensional space described earlier. Taking two points, $J(x_1 + y_1 i)$ and $K(x_2 + y_2 i)$, and using D_2 to evaluate the x and y constants, will give the length of segment JK.

6 Python Implementation

Now that we know the formulas, they can be adapted into Python functions.

```python
def one_d(p1, p2):              # define a function that takes two arguments
    return abs(p1 - p2)         # return the absolute difference (1D)

def two_d(p1, p2):              # for ‘typical’ coords and complex constants
    x = (p1[0] - p2[0]) ** 2    # take values from respective indices
    y = (p1[1] - p2[1]) ** 2    # (separated terms for aesthetics)
    return (x + y) ** (1 / 2)   # return 2D distance

def three_d(p1, p2):
    x = (p1[0] - p2[0]) ** 2
    y = (p1[1] - p2[1]) ** 2
    z = (p1[2] - p2[2]) ** 2
    return (x + y + z) ** (1 / 2) # return 3D distance
```

```python
print(one_d(8,13))
print(two_d([4,-1],[-3, 11]))
print(three_d([10,9,6],[-4,-22,2]))

# Output:
5
13.892443989449804
34.249087579087416
```