Activity 1

The System of ODEs

$$\frac{dx_1}{dt} = -\frac{1}{2\pi}\frac{\Gamma_2(y_1 - y_2)}{d_{12}^2}$$

$$\frac{dy_1}{dt} = \frac{1}{2\pi}\frac{\Gamma_2(x_1 - x_2)}{d_{12}^2}$$

$$\frac{dx_2}{dt} = -\frac{1}{2\pi}\frac{\Gamma_1(y_2 - y_1)}{d_{12}^2}$$
$$\frac{dy_2}{dt} = \frac{1}{2\pi}\frac{\Gamma_1(x_2 - x_1)}{d_{12}^2}$$

Solving these

It is clear to see:
$$\Gamma_1\frac{dx_1}{dt} = -\Gamma_2\frac{dx_2}{dt}$$
$$\Gamma_1\frac{dy_1}{dt} = -\Gamma_2\frac{dy_2}{dt}$$

Integrate both sides with respect to t, to get:

1. $\Gamma_1 x_1 = -\Gamma_2 x_2 + \alpha$
2. $\Gamma_1 y_1 = -\Gamma_2 y_2 + \beta$

We can now see, we have:
$$x_1 - x_2 = -\left(1 + \frac{\Gamma_2}{\Gamma_1}\right)x_2 + \alpha = -\left(\frac{\Gamma_1 + \Gamma_2}{\Gamma_1}\right)x_2 + \frac{\alpha}{\Gamma_1}$$

Similarly:
$$y_1 - y_2 = -\left(\frac{\Gamma_1 + \Gamma_2}{\Gamma_1}\right)y_2 + \frac{\beta}{\Gamma_1}$$

We can use these observations to now solve for (x_2, y_2). Thankfully, the d_{12} term is eliminated.

$$\frac{dy_1}{dx_1} = \frac{\frac{dy_1}{dt}}{\frac{dx_1}{dt}} = -\frac{x_1 - x_2}{y_1 - y_2} = -\frac{-(\Gamma_1 + \Gamma_2)x_2 + \alpha}{-(\Gamma_1 + \Gamma_2)y_2 + \beta}$$

We rearrange this, and then integrate with respect to x_2 on both sides:

$$\int -(\Gamma_1 + \Gamma_2)y_2 + \beta dy_1 = -\int -(\Gamma_1 + \Gamma_2)x_2 + \alpha dy_1$$
$$-\tfrac{1}{2}(\Gamma_1 + \Gamma_2)y_2^2 + \beta y_2 + C = -\left[-\tfrac{1}{2}(\Gamma_1 + \Gamma_2)x_2^2 + \alpha x_2\right]$$

Rearranging, we finally get:

3. $\frac{1}{2}(\Gamma_1 + \Gamma_2)(x_2^2 + y_2^2) - \alpha x_2 - \beta x_2 = C$

We can now spot that $\frac{dy_1}{dx_1} = \frac{dy_2}{dx_2}$, and that the relationships between (x_1, y_1) and (x_2, y_2) have a symmetrical form:

$\Gamma_2 x_2 = -\Gamma_1 x_1 + \alpha$
$\Gamma_2 y_2 = -\Gamma_1 y_1 + \beta$

Hence, we can just swap 1 by 2 in our final equation above to get:

4. $\frac{1}{2}(\Gamma_1 + \Gamma_2)(x_1^2 + y_1^2) - \alpha x_1 - \beta x_1 = D$

What's left now is to find the values of the constants.
Substituting our initial conditions in, we get:

$\alpha := \Gamma_1 a_1 + \Gamma_2 a_2$
$\beta := \Gamma_1 b_1 + \Gamma_2 b_2$
$C := \frac{1}{2}(\Gamma_1 + \Gamma_2)(a_2^2 + b_2^2) - \alpha a_2 - \beta b_2$
$D := \frac{1}{2}(\Gamma_1 + \Gamma_2)(a_1^2 + b_1^2) - \alpha a_1 - \beta b_1$

You can simplify these a little if you'd like, but it's not necessary.

What are the cases?

If $\Gamma_1 = -\Gamma_2$ then the squared terms disappear, and we get two lines, just with different y-intercepts, C and D (hence parallel lines).

If $\Gamma_1 = \Gamma_2$, we use 1 and 2 to write $x_2^2 + x_1^2$ and substitute it into 3. We make some more substitutions, and can follow through to get $C = D$. Since all the other coefficients are the same, we can conclude that the circles traversed are the same.

If $\Gamma_1 \neq \Gamma_2$, we just have different circles.

Activity 2

```
def getField(self, other_pos):
        """Get the velocity induced at other_pos by the vortex, as a tuple."""
        x, y = other_pos
        self_x, self_y = self.pos
        dsquared = (x - self_x) ** 2 + (y - self_y) ** 2
        if dsquared == 0:
                return (0, 0)
        else:
                return (
                        -self.circulation * (y - self_y) / dsquared,
                        self.circulation * (x - self_x) / dsquared,
                )

def moveVortex(self, vortexArray, timeStep):
        """Compute the velocity of the vortex, and move it."""
        # timeStep is how many times we move with the same velocity
        # each time the method is called.
        velocity = (0, 0)
        for otherVortex in vortexArray:
                velocity = velocity + np.array(otherVortex.getField(self.pos))
        self.pos = self.pos + timeStep * np.array(velocity)
```

Things that are likely to go wrong:

- Not dealing with the case when the distance is 0 in `getField` (or alternatively, not excluding the vortex in question from the velocity calculation in `moveVortex`)
- Not using numpy arrays to add arrays pointwise.
- Potentially not using `getField` in `moveVortex`?

Teaching Python from scratch

- Use as a calculator: `+, -, *, **, /`
- Print things
- Variables
    - Strings
    - Ints
    - Numbers etc.
- List, accessing items etc.

```
numbers = [1,4,5,2,1,1,1]
numbers[0] # 1
```

- If statements

```
if x == 0:
        x = 1
else:
        x = 0
```

- For loops, and how it works with any iterator

```
for i in range(10):
        print("hi")

for number in numbers:
        print(number)
```

- Functions (input and output)

```
def average(x,y):
        return (x+y)/2
```

- Indentation is the only thing that matters.
- Infers types + how to handle.
- Numpy, and arrays (more efficient; act like vectors, so pointwise)

```
np.array((1,2)) + np.array((3,4)) # (4,6)
(1,2) + (3,4) # (1,2,3,4)
```

- Note some errors e.g. syntax, dividing by zero
- Ask us if we're available, but also, Google is your friend!
- Teaching the basics of classes

Teaching the basics of classes

We can create an instance of the class, specifying the parameters found in `__init__`.

```
some_vortex = Vortex((0,0), 1)
```

We can access properties, or call functions specific to that class, called methods, like so:

```
some_vortex.color
some_vortex.getField((0,0))
...
```

Each object acts as a member of the class, with the same property names and functions, but with different values.

(Use typical examples of classes to explain the theory, and then move on to showing how to do these things in python.)