Exploring the Behaviour of Multiple Vortices and Hexagonal Lattice Formations

by Hayley Bishop

MAS8091

Supervisor - Carlo Barenghi

1st May 2014

Abstract

Vortices can be found in our everyday lives, from tornadoes to stirring a cup of tea. This report discusses simpler idealised vortices in 2D and how they can affect the movements of all the other vortices around them in a plane. I start by looking at how time stepping methods are used to numerically solve ordinary differential equations. I then use this to replicate the behaviour of two vortices and consider their predictability. I move on to modelling a system of many vortices and observing its chaotic behaviour. Finally I investigate hexagonal lattice formations. I will be using Fortran 95 to simulate the vortex systems.

Contents

Chapter 1

Introduction

1.1 Definition

A vortex is defined as a region within a fluid where the flow is a spinning motion about an imaginary axis, which can be either straight or curved. Vortices can form in fluids such as liquids, gases and plasmas.

1.1.1 Vorticity

The vorticity is a vector that describes the spinning motion at a point in the fluid, as it would be seen by an observer who is moving along with the fluid. Imagine placing a tiny ball, free to move with the fluid, at the point we wanted to look at, and examining how it rotates about its centre. The vorticity, denoted $\vec{\omega}$ is defined as the curl of the velocity field, $\vec{v}$, of the fluid.

$$\begin{aligned}\vec{\omega} &= \nabla \times \vec{v} \\ &= \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \times (v_x, v_y, v_z) \\ &= \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right).\end{aligned}$$

In two dimensions, the vorticity is parallel to the z axis, so there is no z

component and the vorticity can be written as

$$\begin{aligned}\vec{\omega} &= \nabla \times \vec{v} \\ &= \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \times (v_x, v_y) \\ &= \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}.\end{aligned}$$

1.1.2 Types of Vortex

One type of vortex is a wingtip vortex. Wingtip vortices are circular patterns of rotating air left behind a wing as it is creating lift. When a wing generates aerodynamic lift, the air on the top surface has lower pressure relative to the bottom surface. The air flows from below the wing, out around the edge of the wing to the top of the wing in a circular fashion, which creates the vortex. One wingtip vortex trails from the tip of each wing, in opposite directions.

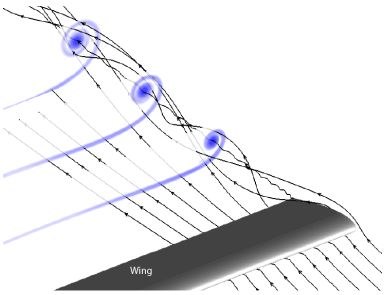

Figure 1.1: A diagram of an aeroplane wing showing how the air flow over the wing creates a wingtip vortex. (Source: http://en.wikipedia.org/wiki/File:Tip_vortex_rollup.png)

Another type of vortex is a point vortex. A point vortex is an idealised vortex in two dimensions so we assume no change in the z direction. Figure 1.2 shows a point vortex located at the origin on the xy plane, with the flow of fluid depicted by the arrows, in this case in an anti clockwise direction. The fluid will move around the point vortex but the vortex cannot move itself. In this project, I will only be considering point vortices.

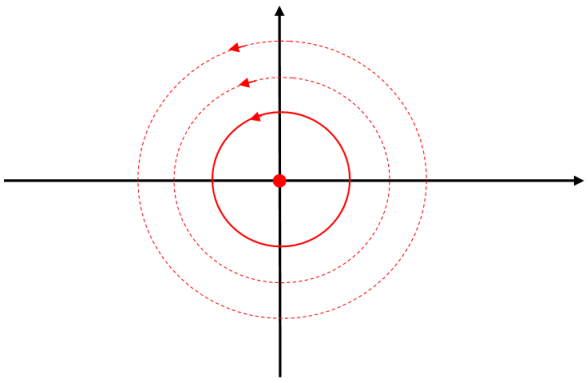

Figure 1.2: A point vortex located at the origin on the xy plane rotating anti-clockwise.

1.1.3 Rotational Vortices

For rotational vortices, the fluid rotates like a rigid body. That is, the particle speed v increases proportionally to the distance r. This means that the further out in the flow we get, the faster the speed. We can see this in Figure 1.3, which shows two balls rotating anticlockwise in the fluid around a vortex. The outer ball is moving faster than the inner one, as it is further away. A rotational vortex can be maintained for an indefinite amount of time in that state through the application of some extra force that is not generated by the fluid motion itself. For example, if a water bucket is spun at constant angular speed about its vertical axis, the water will eventually rotate in rigid-body fashion. In this situation, the rigid rotating enclosure provides an extra force, in this case, an extra pressure gradient in the water, directed inwards, that prevents the flow from becoming irrotational.

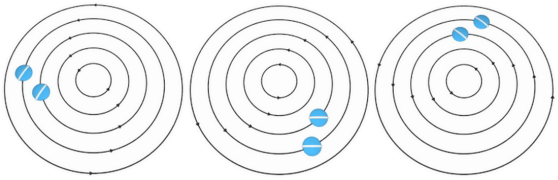

Figure 1.3: This picture shows 2 balls in the flow around a rotational vortex. (Source: http://en.wikipedia.org/wiki/File:Rotational_vortex.gif)

1.1.4 Irrotational Vortices

For irrotational vortices, the particle speed v is inversely proportional to the distance r. This means that the further out in the flow we get, the slower the speed. As we can see from Figure 1.4, the outer ball is moving slower than the inner one, as it is further out. In this case, with the absence of external forces, a vortex usually evolves fairly quickly towards an irrotational flow. For that reason, irrotational vortices are also known as free vortices. However, the ideal irrotational vortex flow is not physically realisable.

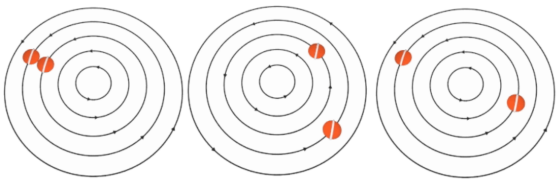

Figure 1.4: This picture shows 2 balls in the flow around an irrotational vortex. (Source:http://en.wikipedia.org/wiki/File:Irrotational_vortex.gif)

1.1.5 Vortex Pairs

A vortex-vortex pair is made up of two close vortices with the same circulation. In Figure 1.5, we have one point vortex located at the red dot and another located at the blue dot with the same circulation. Due to the mutual interaction of their circulating velocity fields, the vortices rotate about a point halfway between the two and they push each other round in a circle.

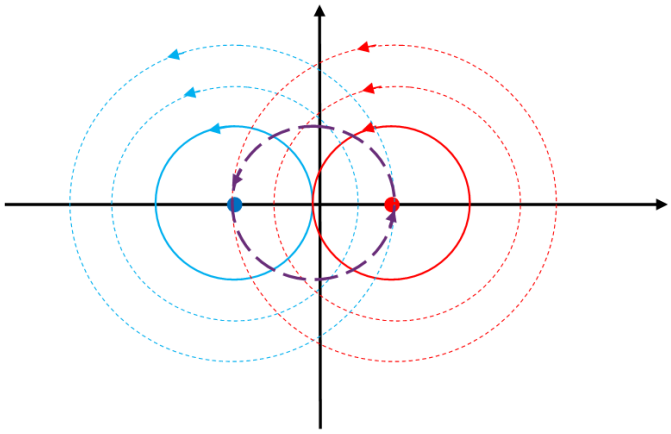

Figure 1.5: This diagram shows one vortex located at the blue dot, with its respective vortex flow depicted by the blue arrows and simiarly another vortex in red. Their combined vortex trail is shown by the dashed purple line.

A vortex-antivortex pair is made up of two close vortices with opposing circulations. Looking at Figure 1.6, we have a red vortex and blue vortex. This time the circulations of the two vortices are in opposite directions, therefore the pair move in a straight line with constant, self induced speed. As the vortices in this case are flowing in towards each other, they push each other down, which is shown by the purple arrows.

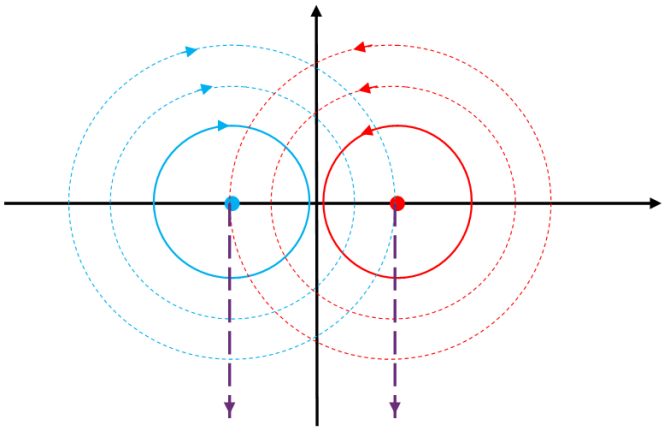

Figure 1.6: This diagram shows an anti-vortex located at the blue dot, with its respective vortex flow depicted by the blue arrows and a vortex which has the opposite circulation in red. Their vortex trails are shown by the dashed purple line.

1.2 Examples

Tornadoes and whirlpools are vortices, with the fluid being air and water respectively. Stirring a cup of tea creates a vortex with the fluid being the tea, but this dissipates quickly when you stop stirring as there is no more energy being put in. Hurricanes are examples of vortices and because of their large scale, hurricanes can be considered as point vortices as their height is such a small proportion of their width so we think of it as lying in a 2D plane.

Planes and formula 1 cars create wingtip vortices. Migratory birds often use each other's wingtip vortices by flying in a V formation. This would mean that all but the leader are flying in the vortex trail from the wing of the bird ahead. This means the bird has to put less effort in to support its own weight, so they can fly for longer.

1.3 Assumptions

In order to make calculations possible, we must make some basic assumptions.

- The fluid flow is two dimensional - The basic flow pattern and characteristics of the motion of the fluid are basically the same as in any parallel plane. This allows us to focus on a single plane which we take to be xy plane. We assume that this plane is the same for the whole of the z axis so we can think of the xy plane to be a cross section of an infinite cylinder.
- The flow is stationary - The velocity of the fluid at any point depends only on the position (x, y) and not on time.
- The fluid is incompressible - The density, or mass per unit volume of the fluid is constant.
- The fluid is non-viscous - A fluid is non-viscous if it has no internal friction. Viscosity is due to the friction in a fluid between nearby particles that are moving at different velocities.
- The circulation can only be $+1$ or -1.

1.4 Finding the Velocity

We start with a vortex located at the origin on the xy plane. We pick a point (x, y) somewhere in the fluid at a distance r from the vortex. Let V be the

tangential velocity, and u and v be the velocities in the x and y directions respectively.

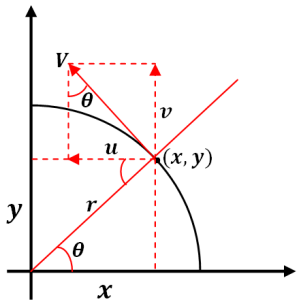

Figure 1.7: A diagram showing how the values for u and v are found.

Using trig ratios, we find

$$u = -V\sin\theta, \qquad v = V\cos\theta.$$

We have to take u to be negative as the x direction is defined to be the right, which is positive, and u is in the opposite direction so we need the minus sign. Next, the polar coordinates for x and y are

$$x = r\cos\theta, \qquad y = r\sin\theta,$$

where $r = \sqrt{x^2 + y^2}$. Rearranging these, we get

$$\cos\theta = \frac{x}{r}, \qquad \sin\theta = \frac{y}{r}.$$

We can sub these back into the equations for u and v to get

$$u = -\frac{Vy}{r}, \qquad v = \frac{Vx}{r}.$$

Finally, we replace V with the formula for the tangential velocity emerging from the point vortex, which is

$$V = \frac{k}{2\pi r}.$$

This gives us the final equations

$$u = -\frac{ky}{2\pi r^2}, \qquad v = \frac{kx}{2\pi r^2},$$

where r is the radius of the vortex and k is the strength of the vortex.

The equations we found for u and v are only valid for when the vortex is situated at the origin. As this isn't usually the case, we need to perform a shift. Let the position of the vortex be given by (x_i, y_i), then our equations for u and v become

$$u = -\frac{k(y - y_i)}{2\pi r^2}, \qquad v = \frac{k(x - x_i)}{2\pi r^2},$$

where $r = \sqrt{(x - x_i)^2 + (y - y_i)^2}$.

For two or more vortices, we calculate the velocity at a point by finding u and v as before, but now we must sum over all vortices, so our equations for u and v become

$$u = -\frac{1}{2\pi} \sum_{i=1}^{N} \frac{k_i(y - y_i)}{r^2}$$

$$v = \frac{1}{2\pi} \sum_{i=1}^{N} \frac{k_i(x - x_i)}{r^2}$$

where N is the total number of vortices.

If we want to find the velocity at a vortex point, to find, for example, how it will move given the effect of all the other vortices, we sum over all vortices except the one that we are looking at. Let (x_j, y_j) be the coordinates of the vortex point where we want to find the velocity. The equations for u and v become

$$u = -\frac{1}{2\pi} \sum_{i=1, i\neq j}^{N} \frac{k_i(y_j - y_i)}{r^2},$$

$$v = \frac{1}{2\pi} \sum_{i=1, i\neq j}^{N} \frac{k_i(x_j - x_i)}{r^2},$$

where $r = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$.

Chapter 2

Time Stepping Methods

2.1 The Euler Method

2.1.1 The Euler Method

The Euler method, named after Leonhard Euler, is a numerical method which is used for solving ordinary differential equations (ODEs) given an initial value. It is known to be the most basic method of numerical integration of ODEs and is the simplest of the Runge-Kutta methods. The Euler method is a first order method of numerical integration. This means that the error per step is proportional to the square of the step size, and the error at a given time is proportional to the step size. I have started by looking at the Euler method as it provides a basis on which to construct more complicated methods.

2.1.2 Informal Geometrical Description

Consider the problem: we are trying to determine the shape of an unknown curve. We are given the point at which the curve starts and the differential equation that it satisfies. This differential equation is the formula by which the gradient of the tangent line to the curve can be computed at any point on the curve, once the position of that point has been found. We know the starting point of the curve, let us call this x_0. Now using the differential equation, the gradient of the curve at x_0 can be calculated and therefore a tangent line can be drawn. Next we take a small step up this tangent line to a point x_1. As we haven't moved very far, x_1 will still be close to the unknown curve. If we pretend that x_1 is actually on the curve, we can use the same method we used to find the gradient for x_0 and therefore find a tangent line at x_1. After several steps like this, a polygonal curve $x_0, x_1, x_2, x_3 \ldots$ is

constructed. This curve does not differ too much from the original curve so we can use it as an approximate solution, as shown in Figure 2.1. The error between the two curves can be decreased by making the step size smaller.

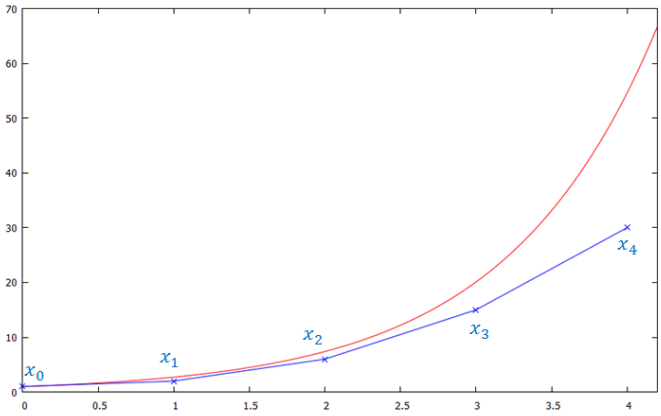

Figure 2.1: The red line shows the exact solution of an ODE. The blue line shows the numerical approximation.

2.1.3 The Method

Given the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = f(t, y(t)), \qquad y(t_0) = y_0.$$

We want to approximate the solution. We choose a step size h and set $t_n = t_0 + nh$ where n is the total number of steps. One step of the Euler method from t_n to $t_{n+1} = t_n + h$ is

$$y_{n+1} = y_n + hf(t_n, y_n).$$

The value of y_n is an approximation of the solution to the ODE at time t_n.

2.1.4 Example

Given the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = yt, \qquad y(0) = 1.$$

We will use the Euler method to approximate the value of $y(2)$. Set the step size $h = 0.5$. The Euler method is $y_{n+1} = y_n + hf(t_n, y_n)$. In this differential

equation, the function f is defined by $f(t, y) = ty$. We start by finding $f(t_0, y_0)$ and from the initial conditions we have $t_0 = 0$ and $y_0 = 1$ so

$$f(t_0, y_0) = f(0, 1) = 0 \times 1 = 0.$$

Then using the Euler method for the first step we get

$$y_1 = y_0 + hf(t_0, y_0) = 1 + 0.5 \times 0 = 1.$$

We can now work out the next few terms:

$$\begin{aligned} y_2 &= y_1 + hf(t_1, y_1) = 1 + (0.5 \times 0.5 \times 1) = 1.25, \\ y_3 &= y_2 + hf(t_2, y_2) = 2 + (0.5 \times 1 \times 1.25) = 1.875, \\ y_4 &= y_3 + hf(t_3, y_3) = 6 + (0.5 \times 1.5 \times 1.875) = 3.28125. \end{aligned}$$

Using the Euler method, we get the answer $y_4 = 3.28125$. The exact solution of the differential equation is $y(t) = e^{\frac{t^2}{2}}$. So the exact solution at $t = 2$ is $y(2) = e^2 \approx 7.389$. This gives a 55.6% error which is not a very good approximation.

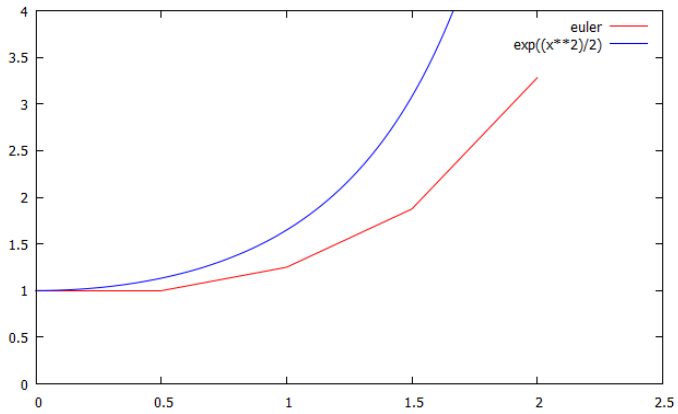

Figure 2.2: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Euler method using a step size of $h = 0.5$.

As mentioned earlier, decreasing the step size h will make the approximation more accurate. Let us try the same problem with a step size of 0.1. Using the code in Appendix A, we find $y(2) \approx 5.97323$ using this step size. Now we have an error of 19.2%. If we keep decreasing the step size, the approximation becomes more accurate. See table.

h	$y(2)$	% Error
0.5	3.281	55.6
0.1	5.973	19.2
0.01	7.220	2.3

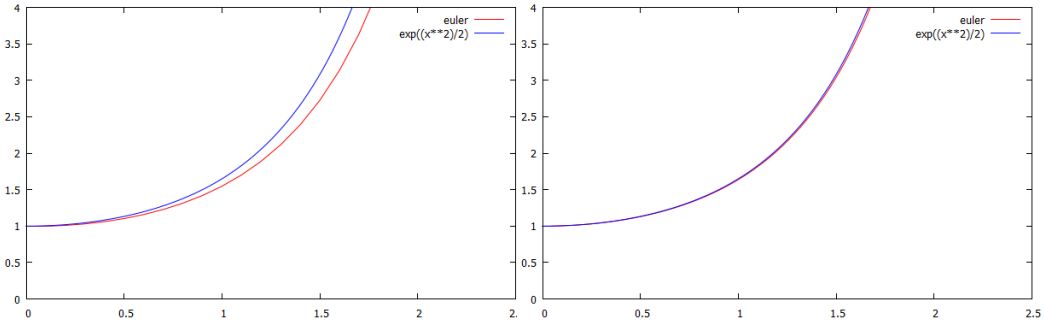

Figure 2.3: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Euler method. The left graph uses a step size of $h = 0.1$ and the right graph using a step size of $h = 0.01$.

2.2 The Runge-Kutta Methods

The Runge-Kutta methods are an important family of iterative methods used for the approximation of solutions of ordinary differential equations. They are single step methods with multiple stages per step. They do not require derivatives of the function $f(t, y)$ as the Taylor methods do, and are therefore general purpose initial value problem solvers. These techniques were developed by C. Runge and M. W. Kutta. The most common of these methods is the Runge-Kutta method of order 4, also known as RK4 or the classical Runge-Kutta method. I will also be looking at a simplified version of this, the Runge-Kutta method of order 2.

2.2.1 The Fourth Order Method

The Method

Given the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = f(t, y), \qquad y(t_0) = y_0.$$

We want to approximate the solution. We choose a step size h and set $t_n = t_0 + nh$ where n is the total number of steps. One step of the RK4

method from t_n to $t_{n+1} = t_n + h$ is

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4),$$

where

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_1), \\
k_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_2), \\
k_4 &= f(t_n + h, y_n + hk_3).
\end{aligned}$$

Here y_{n+1} is determined by the current value y_n and the weighted average of four increments. k_1 is the increment based on the slope at the beginning of the interval, k_2 and k_3 are the increments based on the slope at the midpoint of the interval and k_4 is the increment based on the slope at the end of the interval. When averaging the four increments, greater weight must be given to the increments at the midpoint.

Example

Let us look at the example we had in the previous section:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = yt, \qquad y(0) = 1.$$

As before we shall start with a step size $h = 0.5$. Using the code in Appendix B, we get

$$\begin{aligned}
y_1 &= 1.13314, \\
y_2 &= 1.64853, \\
y_3 &= 3.07798, \\
y_4 &= 7.36680.
\end{aligned}$$

The exact solution is the same as before, $y(2) \approx 7.389$. Using the new value for y_4, this gives an error of 0.3% with the step size of $h = 0.5$.

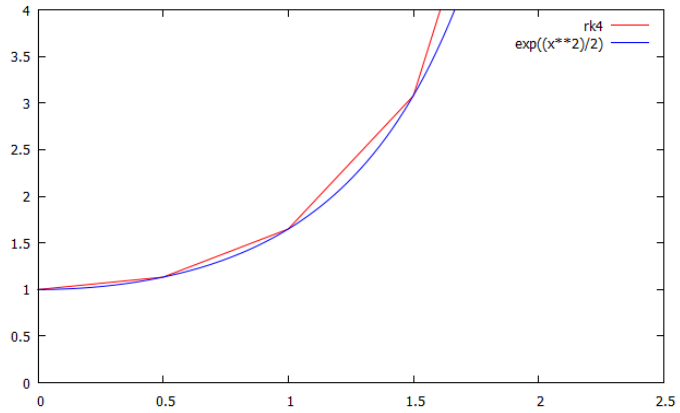

Figure 2.4: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Runge-Kutta method of order 4 using a step size of $h = 0.5$.

As before, decreasing the step size will make the approximation more accurate. See table.

h	y_4	% Error
0.5	7.3668	0.3
0.1	7.3890	0
0.01	7.3890	0

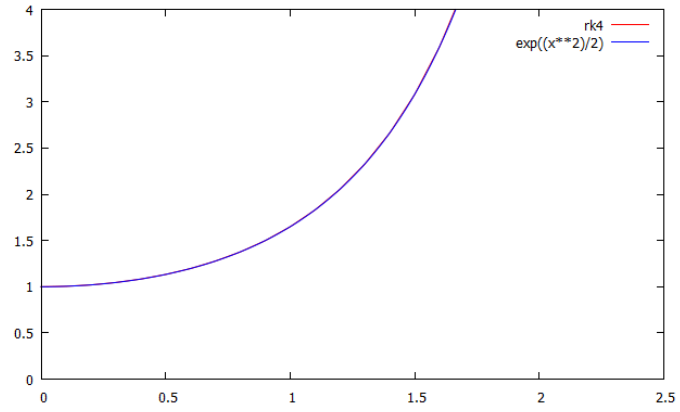

Figure 2.5: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Runge-Kutta method of order 4 using a step size of $h = 0.1$. The graph of $h = 0.01$ looks exactly the same as this.

2.2.2 The Second Order Method

The Method

Given the initial value problem

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = f(t, y), \qquad y(t_0) = y_0.$$

We want to approximate the solution. We choose a step size h and set $t_n = t_0 + nh$ where n is the total number of steps. One step of the RK2 method from t_n to $t_{n+1} = t_n + h$ is

$$y_{n+1} = y_n + h(1/2k_1 + 1/2k_2),$$

where

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + h, y_n + hk_1). \end{aligned}$$

As before, y_{n+1} is determined by the current value y_n but this time only uses the weighted average of two increments. k_1 is the increment based on the slope at the beginning of the interval and k_2 is the increment based on the slope at the end of the interval.

Example

As before, we will use the differential equation $\frac{\mathrm{d}y}{\mathrm{d}t} = y'(t) = yt$ with the initial condition $y(0) = 1$ and we shall start with a step size $h = 0.5$. Using the code in Appendix C, we get

$$\begin{aligned} y_1 &= 1.125, \\ y_2 &= 1.58203, \\ y_3 &= 2.71912, \\ y_4 &= 5.60818. \end{aligned}$$

The exact solution is the same as before, $y(2) \approx 7.389$, which gives an error of 24% with the step size of $h = 0.5$.

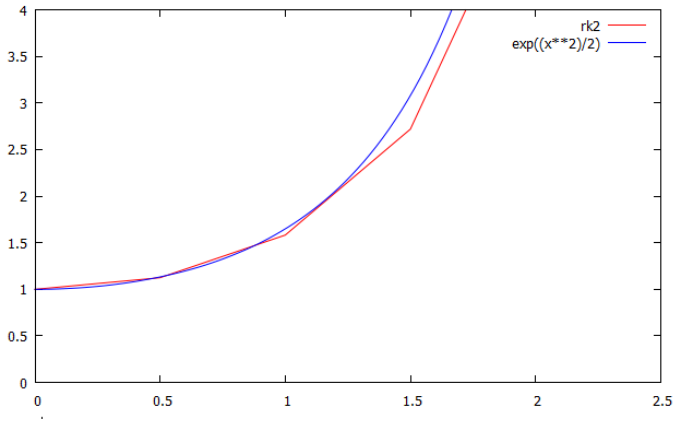

Figure 2.6: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Runge-Kutta method of order 2 using a step size of $h = 0.5$.

As before, decreasing the step size will make the approximation more accurate. See table.

h	y_4	% Error
0.5	5.6082	24
0.1	6.6209	10.4
0.01	7.2931	1.3

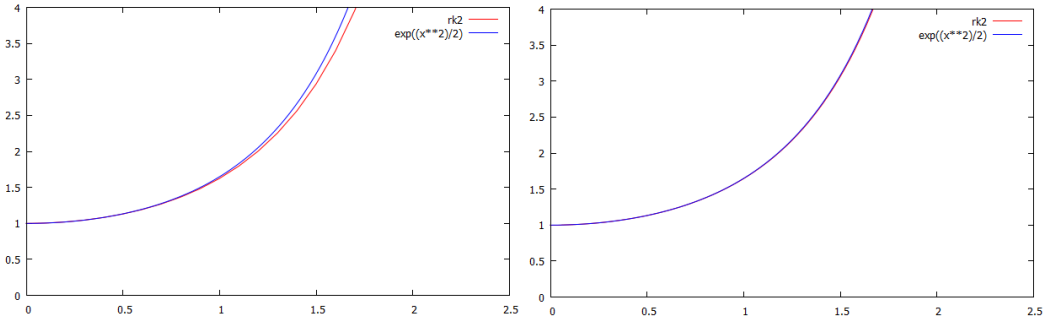

Figure 2.7: The blue line shows the exact solution of an ODE. The red line shows the numerical approximation using the Runge-Kutta method of order 2. The left graph uses a step size of $h = 0.1$ and the right graph using a step size of $h = 0.01$.

Chapter 3

Vortices

3.1 Two Vortices

I wrote a Fortran program, attached in Appendix D, to simulate the movement of two vortices. It takes the current position of the vortex and moves it according to the velocity experienced at that position due to the flow of the other vortex. As with the examples in the time stepping methods chapter, I started with the Euler method and worked up to using the Runge-Kutta method of order 4. I used a time step of $h = 0.001$ and plotted the position of the vortices at each time step.

3.1.1 Vortex-Vortex Pair

I talked briefly in chapter 1 about how we would expect a vortex-vortex pair to behave. I placed one vortex at $(1, 0)$ and the other at $(-1, 0)$ and used a circulation of $+1$ for both vortices, so they are both rotating in the same direction.

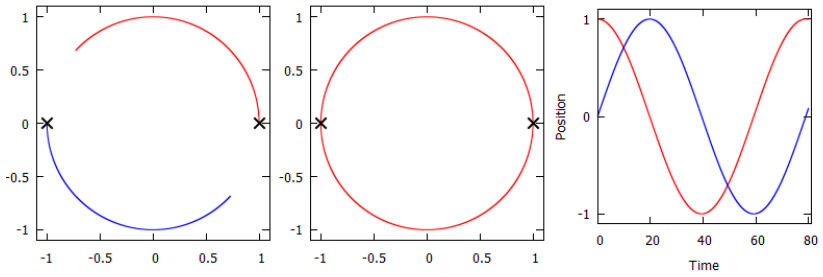

Figure 3.1: The first graph shows partial trails of each vortex, where the initial position of the vortex is located at the ×'s. The second graph is the full trail of the vortices showing how they push each other round in a perfect circle. The final graph show the displacement of a vortex over time. The red line shows the position of the x coordinate and the blue line shows the position of the y coordinate.

As we can see from the first two pictures in Figure 3.1, the vortices behave like we would expect. Due to the fact that they have the same direction of flow, they push each other around in a circle. The last picture in Figure 3.1 shows the position of the x and y coordinates of one of the vortices plotted against time. This was done in order to work out the speed at which the vortices are moving. It does not matter which vortex we look at as they are both moving at the same speed. To work the speed out, we take the time between two peaks on either of the curves, and input this into the formula,

$$\omega_{\text{num}} = \frac{2\pi}{\tau},$$

where ω is the speed and τ is the difference in time between the peaks. This gives a value of $\omega = 0.07956$. As this is just an numerical approximation, we can compare this to the exact value for the speed. Using the formula,

$$\omega_{\text{exact}} = \frac{k}{\pi d^2},$$

where k is the circulation and d is the distance between the two vortices. In this case, k is 1 as both our vortices have a circulation of 1 and the distance is 2. This gives an exact speed of,

$$\begin{aligned} \omega_{\text{exact}} &= \frac{1}{\pi \times 2^2} \\ &= 0.07958. \end{aligned}$$

Next we can work out the percentage error using

$$\begin{aligned}\text{Error} &= \left|\frac{\omega_{\text{num}} - \omega_{\text{exact}}}{\omega_{\text{exact}}}\right| \times 100\% \\ &= 0.02047\%.\end{aligned}$$

We have a percentage error of only 0.02047%. This means that our approximation is very good. I tried a few different values for the time step h to see how much difference having a small time step has on the accuracy of our approximation. As we can see in the table below, the smaller the value of h, the smaller the percentage error and therefore the better the approximation.

h	ω	% Error
0.5	0.07181	9.76376
0.1	0.07795	2.03791
0.01	0.07941	0.20622
0.001	0.07956	0.02047

3.1.2 Vortex-Antivortex Pair

Earlier I talked about a vortex-antivortex pair and how they would propel each other in a straight line. This was confirmed when I placed a vortex at the point $(1, 0)$ with the circulation $k = 1$ and the other at $(-1, 0)$ with the opposite circulation of $k = -1$.

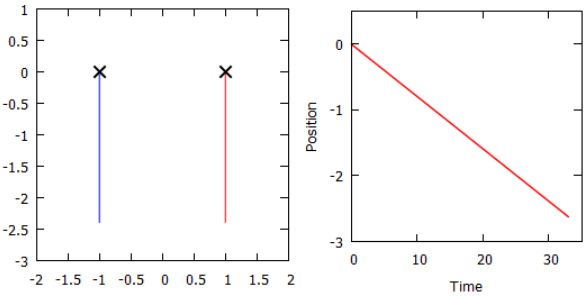

Figure 3.2: The first picture shows the two vortex trails with their initial positions located at the ×'s. The second picture shows the position of the y coordinate of the first vortex over time. The x coordinate does not change so there is no need to plot it.

The vortices are now rotating towards each other and as a result push each other downwards which can be seen in the first picture of Figure 3.2. The second picture in Figure 3.2 shows the position of the one of the vortices

plotted against time. Again it doesn't matter which vortex we look at. This time we measure the displacement over time,

$$v_{\text{num}} = \frac{\text{distance}}{\text{time}},$$

where v is the speed. This gives a value of $v = 0.07939$. As this is just a numerical approximation, we can compare this to the exact value for the speed. Using the formula,

$$v_{\text{exact}} = \frac{k}{2d\pi},$$

where k is the circulation and d is the distance between the two vortices. In this case, we take k to be 1 as I used the data from the vortex at $(1, 0)$ which had the circulation of 1. We could have used the other vortex data and the circulation of -1, this would just give a negative value for the speed which is because of the opposite flow of this vortex. Again the distance between the two vortices is 2. This gives an exact speed of,

$$\begin{aligned} v_{\text{exact}} &= \frac{1}{2 \times 2 \times \pi} \\ &= 0.07958. \end{aligned}$$

Next we can work out the percentage error using

$$\begin{aligned} \text{Error} &= \left| \frac{v_{\text{num}} - v_{\text{exact}}}{v_{\text{exact}}} \right| \times 100\% \\ &= 0.23455\%. \end{aligned}$$

We have a percentage error of only 0.23455%. This means that our approximation is very good. Again I tried a few different values for the time step h to see how much difference having a small time step has on the accuracy of our approximation. As we can see in the table below, the smaller the value of h, the smaller the percentage error with the exception of the last value which increases again slightly. This could be due to rounding errors.

h	ω	% Error
0.5	0.07542	5.2233
0.1	0.07855	1.2942
0.01	0.07946	0.1402
0.001	0.07939	0.2346

I have chosen to use $h = 0.001$ for the rest of the simulations, even though the value of $h = 0.01$ is better in the vortex-antivortex simulation, the difference between the two percentage errors is very small compared to the difference in percentage errors for the vortex-vortex simulation of the same two step sizes.

A point to note is that the closer the vortices are to each other, the faster they will move. The vortex-vortex pair will push each other round faster and the vortex-antivortex pair will propel away faster.

3.2 Multiple Vortices

We have just looked at how two vortices can effect each other and how predictable they are. There are some other specific arrangements of vortices for which we can predict the behaviour. An example of this is an equilateral triangle formation, each vortex with the same circulation. By amending the code in Appendix D for three vortices, we can plot the vortex trail for a triangle formation.

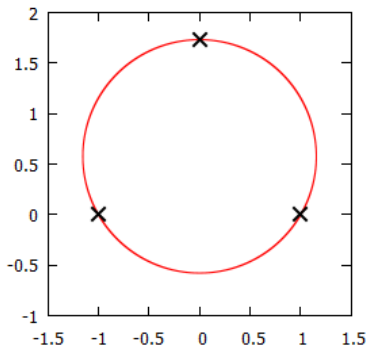

Figure 3.3: This graph shows the vortex trail of three vortices initially placed at the $\times$'s.

Another example is a square formation, where each vortex has the same circulation.

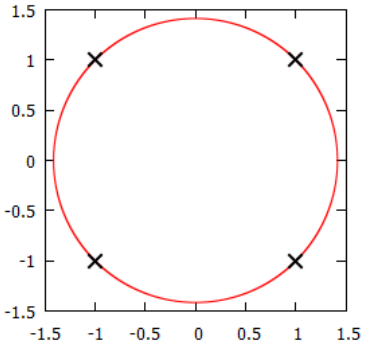

Figure 3.4: This graph shows the vortex trail of four vortices initially placed at the $\times$'s.

This is true for all regular shapes i.e. regular pentagons, hexagons, octagons etc, where each vortex in the arrangement has the same circulation.

3.3 Chaotic Vortices

Most arrangements of many vortices are not predictable. To show this, I chose six random positions for six vortices to sit inside a two by two square. I randomly selected three of the vortices to have a circulation of $+1$ and the other three to have circulation -1. To do this I amended the earlier code for two vortices, now having six vortices and six circulations, see Appendix E. Figure 3.5 shows the movements of the vortices over $150,000$ steps.

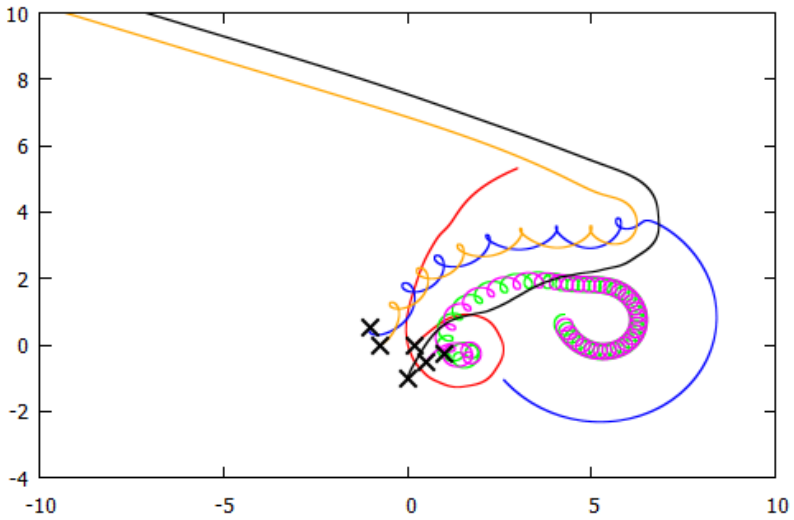

Figure 3.5: This diagram shows six vortex trails, the initial position of each denoted by an $\times$.

Whilst the overall appearance of the graph looks random and chaotic, there are some aspects which replicate what we looked at in section 3.1. For example, the purple and orange vortex trails pair up and shoot off together. This is similar to the vortex-antivortex pairing we saw in section 3.1.2. This happens here because the two vortices become so close and due to their opposing circulations, they propel away in a straight line. As they are moving so fast, they become so far away that the other vortices have no effect on them and they just continue to travel away.

Another display of interesting behaviour is the interaction of the pink and green vortices; they spiral around each other. This must mean they have the same circulation but as this just propels them around each other and not out away from the group, they are still susceptible to the flow of other vortices. This leads to the spiraling and the curving at the end of the trail, which appears to be due to the blue vortex as it follows a similar shape itself.

Another way of showing chaos is by perturbing the formation of vortices we have and seeing how different the trails become. For this I changed the x coordinate of the first vortex from 0.2 to 0.2001, 0.201 and 0.21. As we can see in Figure 3.6, even these small changes can make a big difference. A perturbation of only one coordinate of one vortex by just 0.01 makes a drastic change to the trails of all six vortices.

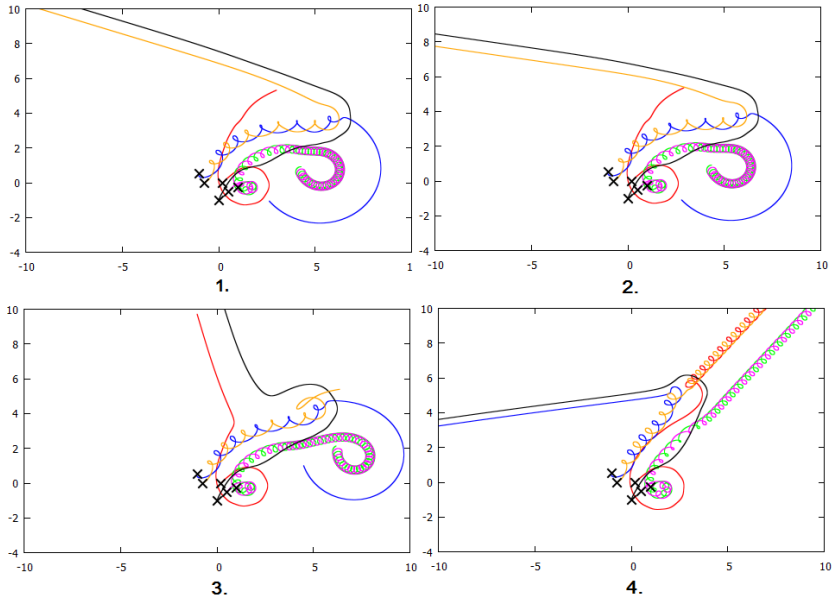

Figure 3.6: Graph 1 shows the original graph where $x = 0.2$. Graph 2 shows the change to $x = 0.2001$. Graph 3 shows the change to $x = 0.201$. Graph 4 shows the change to $x = 0.21$.

I wrote a program, see Appendix F, to find the overall difference between the original graph and a perturbed graph. I started by finding the difference in distance between vortex 1 in the original graph and vortex 1 in the perturbed graph, and repeated this for the rest of the vortices then added all these differences together. Next I took the average of these and took log of the result, and repeated this for each time step. Figure 3.7 shows the differences over time for each of the different perturbed graphs.

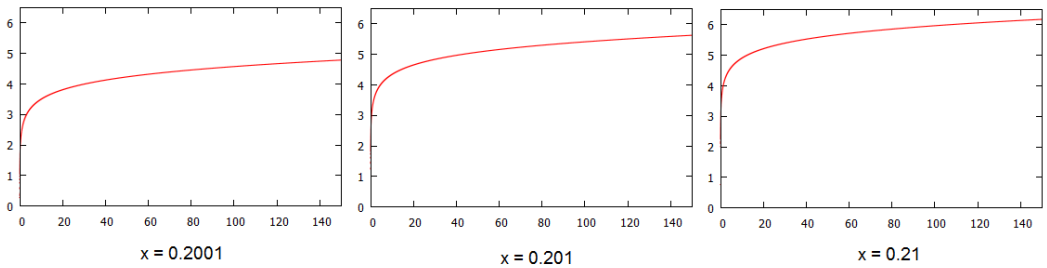

Figure 3.7: These graphs show the difference over time between the points from the original graph and the perturbed graph.

Chapter 4

Lattices

4.1 Definition

A lattice is a regular, periodic configuration of points throughout an area. The most common lattices are equilateral triangle lattices, rhomic lattices, square lattices and parallelogrammic lattices. A lattice is usually the arrangement of molecules in a crystalline solid. The word lattice is also used for regular arrangements of magnetic flux tubes in superconductors and vortex lines in rotating superfluids, such as liquid helium and Bose-Einstein condensates.

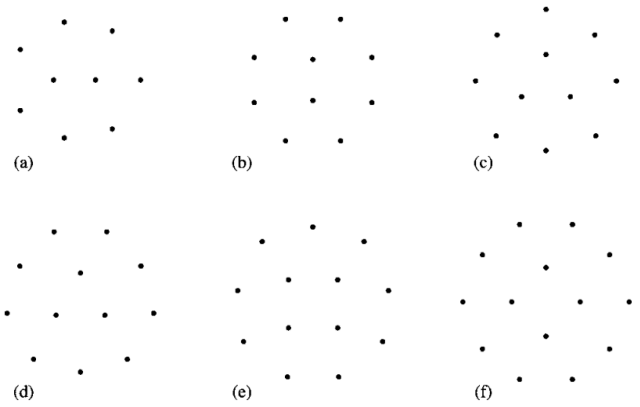

Figure 4.1: These diagrams show just a few different types of lattice formation. (Source: Hassan Aref, 2007, 'Point vortex dynamics: A classical mathematics playground')

4.2 Hexagon

I started with a simple hexagonal lattice. Using a graph of equilateral triangles with side lengths of 1 on top of an xy plane, I was able to work out the coordinates of the corners of a regular hexagon. These become points for six vortices and I included one in the centre of the hexagon at $(0, 0)$, all with a circulation of $+1$.

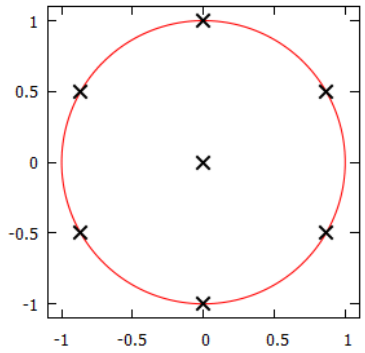

Figure 4.2: This graph shows the vortex trail of these seven vortices in a hexagonal formation with a centre vortex. The initial positions of each vortex are given by $\times$'s.

This arrangement of vortices is similar to those in section 3.2. As we can see in Figure 4.2, the vortices push each other round and their trails fall perfectly on top of each other.

4.3 Double Hexagon

Next I extended the hexagon by adding 12 more points in a hexagon around the first, again each with the same circulation of $+1$, making a hexagonal lattice. As before the vortices push each other round, however this time the trails do not fall perfectly on top of each other.

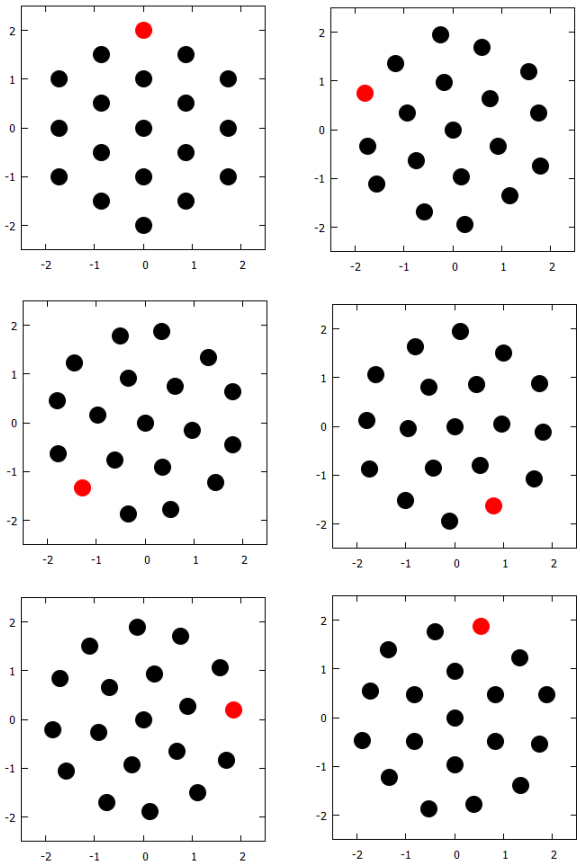

Figure 4.3: This picture shows the position of each of the nineteen vortices at different points in time. I highlighted the top vortex in red and showed its new position in each picture over time. The top left graph being when $t = 0$, then moving across the row to the next time step then continuing down the figure to the bottom right graph which is the final time step which shows almost a complete rotation of the lattice.

The lattice still rotates like a rigid body even though the trails do not fall directly on top of one another. To show this I removed one point from the formation and plotted the new vortex trails. As we can see in Figure 4.4, even though various points have been removed in each of the various pictures, the trails still follow each other round and the overall movement is that of a solid object rotating.

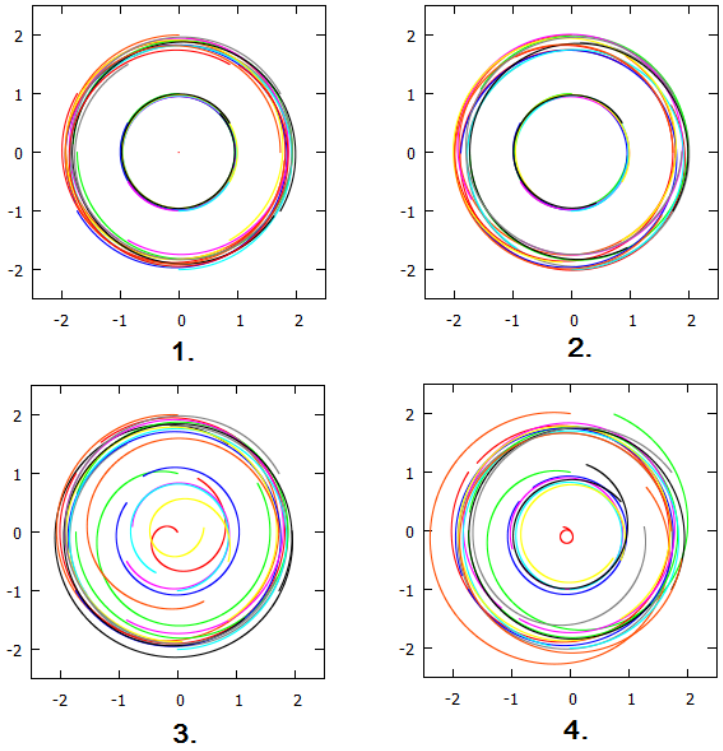

Figure 4.4: Graph 1 is the original graph with all nineteen points. In graph 2, I removed the centre vortex. In graph 3, I removed the top right vortex of the inside hexagon. In graph 4, I removed the top right vortex of the outside hexagon.

Chapter 5

Conclusion

In this report I have looked at how vortices interact with one another, and how the predictability of two vortices comes into play when looking at systems of chaotic vortices. I have also seen how lattices rotate like a rigid body.

In my first meeting with my tutor for this project, he was explaining to me what the project would entail in the form of a similar example of a real vortex that I would be familiar with. Jupiters atmosphere has many vortices interacting with each other which are similar to hurricanes. Jupiters Great Red Spot is counter-clockwise vortex measuring 30,000km east to west, 12,000km north to south, and as the atmosphere is such a thin layer around the planet in comparison to the depth of the planet itself, we can think of it as being on a 2D plane i.e. a point vortex. I could go on to look further into Jupiter's Red Spot and how it relates to what I have looked at.

Another element I could go on to look at would be to enclose the vortices in a finite space. The boundary would prevent the vortices from pairing up and propelling off together indefinitely so it would be interesting to see what happens. I would expect that as the paired vortices would rebound off the boundary back into the rest of the vortices, at some point they would get broken up by another vortex getting closer.

Finally I would have liked to go on to look at other lattice formations to see how similar or how different they are from the hexagonal formation.

.1 Appendix A

```
program euler
!Aim to solve dy/dt = ty = f(t, y) using the Euler method
implicit none
integer ::  nsteps, n
real ::  t, y, told, yold, h, fold
!Output file
open(unit=7, file='euler.dat')
t = 0.0          !Initial t
y = 1.0          !Initial y
h = 0.01          !Time step
nsteps = 200          !Number of steps
write(unit=7, fmt = "(2d17.9)") t, y
do n=1, nsteps          !Time loop
   told = t          !Update old t
   yold = y          !Update old y
   call getsub(told, yold, fold)
   t = told + h          !Get new t
   y = yold + h*fold          !Get new y
   print*, t, y          !Print to screen
   write(unit=7, fmt = "(2d17.9)") t, y
enddo          !Close time loop
close(unit=7)          !Finish
end program euler

subroutine getsub(t, y, f)
!RHS of equation
implicit none
real ::  t, y, f
f = t*y
end subroutine getsub
```

.2 Appendix B

```
program rk4
!Aim to solve dy/dt = yt = f(t, y) using the numerical method
Runge-Kutta order 4
implicit none
integer ::  nsteps, n
real ::  t, y, told, yold, h, k1old, k2old, k3old, k4old
!Output file
open(unit=7, file='rk4.dat')
t = 0.0          !Initial t
y = 1.0          !Initial y
h = 0.01          !Time step
nsteps = 200          !Number of steps
write(unit=7, fmt = "(2d17.9)") t, y
do n=1, nsteps          !Time loop
   told = t          !Update old t
   yold = y          !Update old y
   call getsub(told, yold, h, k1old, k2old, k3old, k4old)
   t = told + h          !Get new t
   y = yold + (h/6)*(k1old+(2*k2old)+(2*k3old)+k4old) !Get new y
   print*, t, y          !Print to screen
   write(unit=7, fmt = "(2d17.9)") t, y
enddo          !Close time loop
close(unit=7)          !Finish
end program rk4

subroutine getsub(t, y, h, k1, k2, k3, k4)
!RHS of equation
implicit none
real ::  t, y, h, k1, k2, k3, k4
k1 = t*y
k2 = (t+(0.5*h))*(y+(0.5*h*k1))
k3 = (t+(0.5*h))*(y+(0.5*h*k2))
k4 = (t+h)*(y+(h*k3))
end subroutine getsub
```

.3 Appendix C

```
program rk2
!Aim to solve dy/dt = yt = f(t, y) using the numerical method
Runge-Kutta order 2
implicit none
integer ::  nsteps, n
real ::  t, y, told, yold, h, k1old, k2old
!Output file
open(unit=7, file='rk2.dat')
t = 0.0          !Initial t
y = 1.0          !Initial y
h = 0.01          !Time step
nsteps = 200          !Number of steps
write(unit=7, fmt = "(2d17.9)") t, y
do n=1, nsteps          !Time loop
   told = t          !Update old t
   yold = y          !Update old y
   call getsub(told, yold, k1old, k2old, h)
   t = told + h          !Get new t
   y = yold + 0.5*(k1old + k2old)          !Get new y
   print*, t, y          !Print to screen
   write(unit=7, fmt = "(2d17.9)") t, y
enddo          !Close time loop
close(unit=7)          !Finish
end program rk2

subroutine getsub(t, y, k1, k2, h)
!RHS of equation
implicit none
real ::  t, y, k1, k2, h
k1 = h*y*t
k2 = h*(y+(h*k1))*(t+h)
end subroutine getsub
```

.4 Appendix D

```
!Aim to find the velocity at a point and use this to move a
vortex, using the numerical method Runge-Kutta order 4
module parameters
!set parameters
implicit none
real, parameter ::  h = 0.1
integer, parameter ::  nsteps = 1500
integer, parameter ::  nv = 2
end module

program vortex
use parameters
implicit none
integer ::  n,skip,i
real, dimension(nv) ::  xv,yv,cv,xvold,yvold,xvcall,yvcall
real ::  u,v,uold,vold,t
!vortex 1
xv(1)=1.0
yv(1)=0.0
!vortex 2
xv(2)=-1.0
yv(2)=0.0
!circulations
cv(1)=1.0
cv(2)=-1.0
n=0
t=n*h
open(unit=7, file='vortex.dat')
do n=1, nsteps
   do i=1,nv
      xvcall=xv
      yvcall=yv
      xvold=xv
      yvold=yv
      skip = i
      call getvelocity(xvcall(i),yvcall(i),&
         &xvold,yvold,cv,skip,uold,vold)
      k1u=uold*h
      k1v=vold*h
```

```
      call getvelocity(xvcall(i)+(0.5*k1v),yvcall(i)+(0.5*k1u),&
         &xvold,yvold,cv,skip,uold,vold)
      k2u=uold*h
      k2v=vold*h
      call getvelocity(xvcall(i)+(0.5*k2v),yvcall(i)+(0.5*k2u),&
         &xvold,yvold,cv,skip,uold,vold)
      k3u=uold*h
      k3v=vold*h
      call getvelocity(xvcall(i)+k3v,yvcall(i)+k3u,&
         &xvold,yvold,cv,skip,uold,vold)
      k4u=uold*h
      k4v=vold*h
      xv(i) = xvold(i)+ k1u/6 + k2u/3 + k3u/3 + k4u/6
      yv(i) = yvold(i)+ k1v/6 + k2v/3 + k3v/3 + k4v/6
   enddo
   t=n*h
   write(unit=7, fmt = "(5d17.9)") t,xv(1),yv(1),xv(2),yv(2)
enddo
close(unit=7)
end program vortex
subroutine getvelocity(x,y,xv,yv,cv,skip,u,v)
!Finding the velocity at a point in the plane
use parameters
implicit none
real ::  u,v,r,vv,pi,x,y,u1,v1
real, dimension(nv) ::  xv,yv,cv
integer ::  j,skip
pi = 4.0*atan(1.0)
u=0.0
v=0.0
do j=1,nv
   if (j/=skip) then
      r = sqrt((x-xv(j))**2+(y-yv(j))**2)
      vv= (cv(j))/(2*pi*(r**2))
      u1 = -vv*(y-yv(j))
      v1 = vv*(x-xv(j))
      u=u+u1
      v=v+v1
   endif
enddo
end subroutine getvelocity
```

.5 Appendix E

```
!Aim to find the velocity at a point and use this to move a
vortex, using the numerical method Runge-Kutta order 4
module parameters
!set parameters
implicit none
real, parameter ::  h = 0.001
integer, parameter ::  nsteps = 150000
integer, parameter ::  nv = 6
end module

program vortex
use parameters
implicit none
integer ::  n,skip,i
real, dimension(nv) ::  xv,yv,cv,xvold,yvold,xvcall,yvcall
real ::  u,v,uold,vold,t,k1u,k2u,k3u,k4u,k1v,k2v,k3v,k4v
!vortex 1
xv(1)=0.21
yv(1)=0.0
!vortex 2
xv(2)=1.0
yv(2)=-0.25
!vortex 3
xv(3)=-1.0
yv(3)=0.5
!vortex 4
xv(4)=0.5
yv(4)=-0.5
!vortex 5
xv(5)=0.0
yv(5)=-1.0
!vortex 6
xv(6)=-0.75
yv(6)=0.0
!circulations
cv(1)=1.0
cv(2)=-1.0
cv(3)=1.0
cv(4)=-1.0
```

```
cv(5)=-1.0
cv(6)=1.0
n=0
t=n*h
open(unit=7, file='vortex.dat')

do n=1, nsteps
   do i=1,nv
      xvcall=xv
      yvcall=yv
      xvold=xv
      yvold=yv
      skip = i
      call getvelocity(xvcall(i),yvcall(i),&
         &xvold,yvold,cv,skip,uold,vold)
      k1u=uold*h
      k1v=vold*h
      call getvelocity(xvcall(i)+(0.5*k1v),yvcall(i)+(0.5*k1u),&
         &xvold,yvold,cv,skip,uold,vold)
      k2u=uold*h
      k2v=vold*h
      call getvelocity(xvcall(i)+(0.5*k2v),yvcall(i)+(0.5*k2u),&
         &xvold,yvold,cv,skip,uold,vold)
      k3u=uold*h
      k3v=vold*h
      call getvelocity(xvcall(i)+k3v,yvcall(i)+k3u,&
         &xvold,yvold,cv,skip,uold,vold)
      k4u=uold*h
      k4v=vold*h
      xv(i) = xvold(i)+ k1u/6 + k2u/3 + k3u/3 + k4u/6
      yv(i) = yvold(i)+ k1v/6 + k2v/3 + k3v/3 + k4v/6
   enddo
   t = n*h
   write(unit=7, fmt = "(12d17.9)") xv(1),yv(1),xv(2),yv(2),&
        &xv(3),yv(3),xv(4),yv(4),xv(5),yv(5),xv(6),yv(6)
enddo
close(unit=7)
end program vortex

subroutine getvelocity(x,y,xv,yv,cv,skip,u,v)
!Finding the velocity at a point in the plane
```

```
use parameters
implicit none
real ::  u,v,r,vv,pi,x,y,u1,v1
real, dimension(nv) ::  xv,yv,cv
integer ::  j,skip
pi = 4.0*atan(1.0)
u=0.0
v=0.0
do j=1,nv
   if (j/=skip) then
      r = sqrt((x-xv(j))**2+(y-yv(j))**2)
      vv= (cv(j))/(2*pi*(r**2))
      u1 = -vv*(y-yv(j))
      v1 = vv*(x-xv(j))
      u = u+u1
      v = v+v1
   endif
enddo
end subroutine getvelocity
```

.6 Appendix F

```
!Aim to find the average difference between the same points
in time on two graphs
module parameters
!set parameters
implicit none
real, parameter ::  h=0.001
integer, parameter ::  nsteps = 150000
integer, parameter ::  nv = 6
end module

program difference
use parameters
implicit none
real, dimension(nv,nsteps) ::  x,y,xx,yy
real ::  d,a,t,l,sum_d,diff
integer ::  i,j

open(unit=7, file='chaos1.dat')
open(unit=8, file='chaos2.dat')
open(unit=9, file='chaos3.dat')
open(unit=10, file='chaos4.dat')
open(unit=11, file='difference.dat')

!read files
do i=1,nv
   read(7,*) (x(i,j),j=1,nsteps)
enddo
do i=1, nv
   read(8,*) (y(i,j),j=1,nsteps)
enddo
do i=1, nv
   read(9,*) (xx(i,j),j=1,nsteps)
enddo
do i=1, nv
   read(10,*) (yy(i,j),j=1,nsteps)
enddo

j=0
t=j*h
```

```
sum_d=0.0

do j=1, nsteps
   do i=1, nv
      a=(x(i,j)-xx(i,j))**2+(y(i,j)-yy(i,j))**2   !difference
between two vortices
      d=sqrt(a)
      sum_d=sum_d+d   !sum of difference between all six vortices
   enddo
   diff=sum_d/nv       !average difference in summmed distance
   l=log10(diff)       !log of average difference
   t=j*h         !time step
   write(unit=11, fmt='(2d17.9)') t,l
   print*, t,l
enddo

close(unit=7)
close(unit=8)
close(unit=9)
close(unit=10)
close(unit=11)
end program difference
```

Bibliography

[1] Hassan Aref, 2007, 'Point vortex dynamics: A classical mathematics playground'

[2] Hassan Aref, 1983, 'Integrable, chaotic and turbulent vortex motion in two-dimensional flows'

[3] D. Bernard, G. Boffetta, A. Celani and G. Falkovich, 2006, 'Conformal invariance in two-dimensional turbulence'

[4] http://en.wikipedia.org/wiki/Vortex

[5] http://en.wikipedia.org/wiki/Wingtip_vortices

[6] Point vortex dynamics in two dimensions, http://www2.ims.nus.edu.sg/Programs/09fluidss/files/Report_GroupB.pdf

[7] Richard L Burden and J Douglas Faires, Numerical Analysis, 9th Edition