AME 499 - Undergraduate Research

A Study of the Mixing Induced by Two Ideal Vortices Determining the Most Effective Vortex Configuration For a Defined Measurement Quantifying Effectiveness

David M. Belczyk

Department of Aerospace and Mechanical Engineering

University of Notre Dame

Notre Dame, Indiana 46556-5637

dbelczyk@nd.edu

under the direction of: Dr. Joseph M. Powers and Dr. Mihir Sen

May 9, 2003

Abstract

Various configurations of a three-dimensional fluid system with a velocity field induced by two ideal line vortices were studied, with the purpose of determining the most effective configuration of those vortices for mixing. The distance, as well as the angle between the vortices, was varied. Vortices were assumed to be of equal strength. A means of quantifying the mixing capability of any given two-vortex system was estimated to be the change in surface area with respect to time of a constant volume sphere allowed to deform under the influence of the vortices. The center of the sphere was always located half-way between the vortices. The induced velocity of a particle was determined numerically. It was found that particle motion was periodic on the surface of a sphere when the two vortices intersected one another. When there was a distance between the vortices, motion was non-periodic. The system that caused the greatest change in surface area with respect to time of a constant volume sphere was one that had a distance between the vortices equal to the diameter of the sphere with the vortices at an angle of $\pi/4$ to one another. In this limited sense, this was the most effective mixing system.

1 Introduction

1.1 Vortices

The system studied was composed of two fixed, ideal line vortices that each induced a counter-clockwise velocity about themselves. A line vortex is, in geometrical terms, a line in three-dimensional space that induces this surrounding velocity [1]. This velocity is known as a flow field because it can be represented by a dimensional distribution of vectors acting on the particle in space. The equation governing the flow field around an ideal vortex is a solution to what is known as the Laplace equation. Deriving this equation shows crucial assumptions in determining the flow field that was built into the numerical simulation. Eq. (1) is the linear momentum equation for compressible, viscous flow,

$$\rho\left[\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{v}\right] = \rho\boldsymbol{f} - \nabla p + \nabla\cdot\boldsymbol{\tau}, \tag{1}$$

where ρ is fluid density, $\boldsymbol{v}$ is fluid velocity, $\boldsymbol{f}$ is the applied body force, p is the pressure, and $\boldsymbol{\tau}$ is the stress tensor. By taking the curl of Eq. (1), defining $\boldsymbol{f}$ as the gradient of a potential function and applying the vector identities that $\nabla\times\nabla x = 0$ and $\nabla\cdot(\nabla\times x) = 0$, Eq. (2) is obtained,

$$\frac{\partial\boldsymbol{\omega}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{\omega} - \boldsymbol{\omega}\cdot\nabla\boldsymbol{v} + \boldsymbol{\omega}\nabla\cdot\boldsymbol{v} = \frac{1}{\rho}\nabla\times(\nabla\cdot\boldsymbol{\tau}), \tag{2}$$

where the vorticity, $\boldsymbol{\omega}$ is defined as $\boldsymbol{\omega} = \nabla\times\boldsymbol{v}$. By defining the fluid as incompressible, it is possible to say $\nabla\cdot\boldsymbol{v} = 0$, a condition of incompressibility. Also, because it is known that, even in steady flow, an individual fluid particle can change its velocity as well as its position in relation to time, we employ the natural derrivative in Eq. (3) with Eq. (2) to give Eq. (4),

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{v}\cdot\nabla, \tag{3}$$

$$\frac{d\boldsymbol{\omega}}{dt} = \boldsymbol{\omega} \cdot \nabla \boldsymbol{v} + \frac{1}{\rho} \nabla \times (\nabla \cdot \boldsymbol{\tau}) \,. \tag{4}$$

For incompressible, Newtonian flow, the last term in Eq. (4) can be simplified to give Eq. (5) [8],

$$\frac{d\boldsymbol{\omega}}{dt} = \boldsymbol{\omega} \cdot \nabla \boldsymbol{v} + \nu \nabla^2 \boldsymbol{\omega}, \tag{5}$$

where ν is the kinematic viscosity. Note from this equation that if initially $\boldsymbol{\omega} = 0$, then the vorticity will remain zero for all time if the flow is inviscid. If it is assumed that this is the case, then it is known that $\nabla \times \boldsymbol{v} = 0$, a condition for irrotational flow. If a velocity potential function ϕ is defined, such that $\boldsymbol{v} = \nabla \times \phi$, and remembering that $\nabla \cdot \boldsymbol{v} = 0$, Eq. (6), Laplace's equation, follows,

$$\nabla^2 \phi = 0. \tag{6}$$

Because of the assumptions made, Laplace's equation must be satisfied for incompressible, irrotational flow, and thus, any solution to this equation necessarily meets these criteria. Eq. (6) is a second order, linear, partial differential equation. The linearity of it allows that superimposed flows that satisfy Eq. (6) have additive velocity terms [2]. In other words, when a fluid particle is under the induced velocity of two incompressible, irrotational flows, the velocity in any direction is the sum of the velocities induced by each. This an important characteristic to a multiple vortex system like the ones simulated here because, as is shown next, the flow field is a solution of Laplace's equation.

The z-component of the vorticity vector in cylindrical coordinates is given here without proof in Eq. (7) [2],

$$\omega_z = \frac{1}{2} \left[\frac{\partial u_\theta}{\partial r} + \frac{1}{r} \left(u_\theta - \frac{\partial u_r}{\partial \theta} \right) \right], \tag{7}$$

where u_θ is the angular velocity around the line of the vortex and u_r is the velocity along the perpendicular distance r, either towards or away from the vortex. By setting ω_z equal

to zero in Eq. (7) for the condition of irrotationality and setting $u_r = 0$ because a vortex only induces a rotational velocity about itself, Eq. (8) is obtained.

$$\frac{\partial u_\theta}{\partial r} + \frac{u_\theta}{r} = 0 \tag{8}$$

Because it is known that the velocity induced on a particle around an ideal vortex is only a function of the particle's perpindicular distance from the line of the vortex, it can be said that u_θ is a function of r but not of θ. As a result, the derivative in Eq. (8) becomes an ordinary derivative. Solving Eq. (8) gives Eq. (9),

$$u_\theta = \frac{k}{r}, \tag{9}$$

where k is a constant for the strength of the vortex which is equal to $\Gamma/2\pi$. Here, Γ is measured in units of length squared per time, and r in units of length. Thus, the induced flow field surrounding a line vortex is Eq. (10),

$$u_\theta = \frac{\Gamma}{2\pi r}, \tag{10}$$

where r is the perpendicular distance between a fluid particle and the line center of the vortex. Eq. (10) is a solution to the Laplace equation [3], meaning that the flow around the vortex is incompressible, irrotational and additive. The flow will remain irrotational for all time. In addition, it can be shown that Bernoulli's equation is applicable anywhere within the flow field of a vortex system [1].

It should be noted that actual vortices have a viscous central core that affects the velocity surrounding its center. For the purposes of this analysis, however, the line vortex is an idealized version of the real with no core. Particles at the center of a line vortex are at a mathematical singularity where velocity is infinite, and these particles never leave the line.

1.2 Chaos

One question regarding the flows generated in the two vortex system was whether or not they would be periodic. Periodicity is the tendency for any particular dynamic system to behave such that it predictably repeats itself. To illustrate this point, one could imagine the periodicity of a sine wave describing the motion of a frictionless pendulum. Chaos is a developing theory which seeks to explain another phenomenon known as non-periodicity, which is the tendency for a system's behavior to never repeat. It was introduced at the beginning of the 1960's and has many applications in nearly all parts of nature [4]. In this case, however, it is most relevant in showing that non-linear dynamic systems, such as this two-vortex system, are not always predictable or periodic.

Non-linear systems can behave stably, in that small perturbations can be dampened. Sometimes perturbations cause a magnification of the effect of a restoring term in the system. The same system, however, if driven differently by changing parameters, can become unpredictable. This type of unpredictable system is known as chaotic. An example of a discrete, non-linear system that behaves this way is given in Eq. (11),

$$x_{n+1} = cx_n(1 - x_n), \tag{11}$$

where c is a constant parameter that drives the system. Fig. 1 and Fig. 2 show iterative solutions of Eq. (11). Fig. 1 shows the behavior of the system for one particular value of c. For this value, the system stabilized and came to equilibrium. Any perturbation, in the form of a change of the initial x value, does not change the stabilizing trend of the system or its ultimate equilibrium value.

In Fig. 2, the same equation was solved iteratively, only this time with a different value for c. The system is chaotic. It will never come to rest at equilibrium and is unpredictable.

The unpredictable nature of the system is due to a condition known as sensitive dependence on initial conditions [4]. This means that tiny perturbations change the behavior of the system in the long run. To demonstrate this point, Fig. 3 shows two iterative solutions

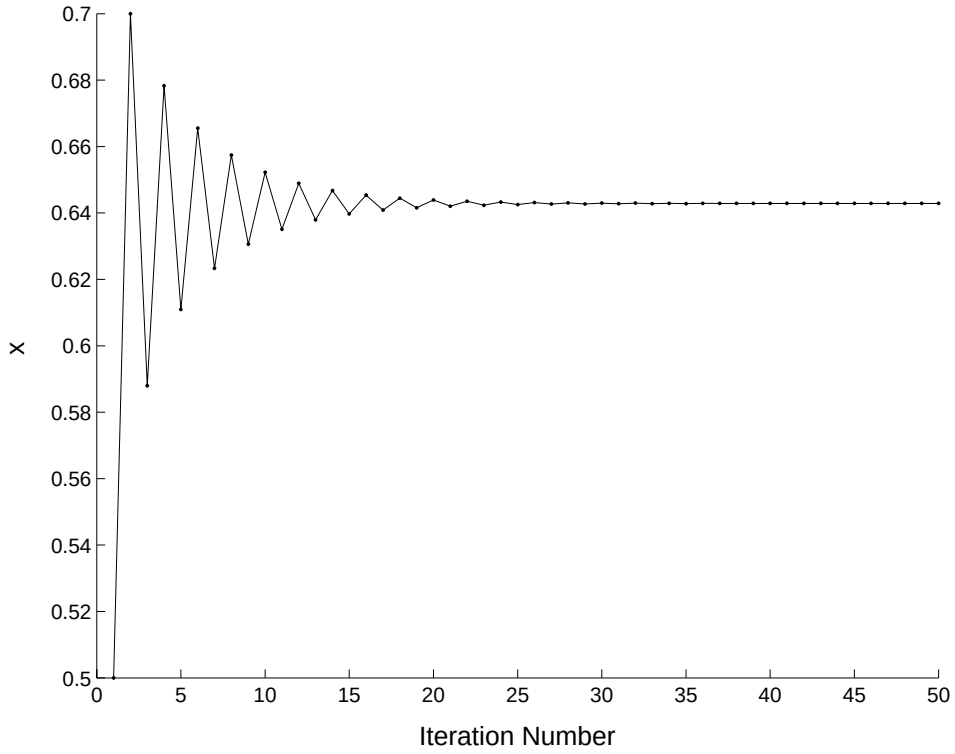

Figure 1: Solution to Eq. (11) with $c = 2.6$.

of Eq. (11) with a change in initial conditions. The two plots initially appear to be the same, but in the long run, the non-linearity of the equation magnifies the tiny perturbation. The plots eventually become two completely different systems.

A final point to be understood regarding chaos theory is that of the strange attractor [4]. Systems that behave chaotically will not map to phase space showing any type of closed loop. In other words, if the system is solved for an infinite amount of time, it will never return to the point at which it started or to any point that has already been a solution. Sometimes, however, regardless of initial conditions, a solution to a chaotic system plotted in phase space will consistently form a similar pattern, although it is never exactly the same. The solution will never repeat, but wind infinitely into the repeating pattern and will always behave differently within the bounds of that pattern.

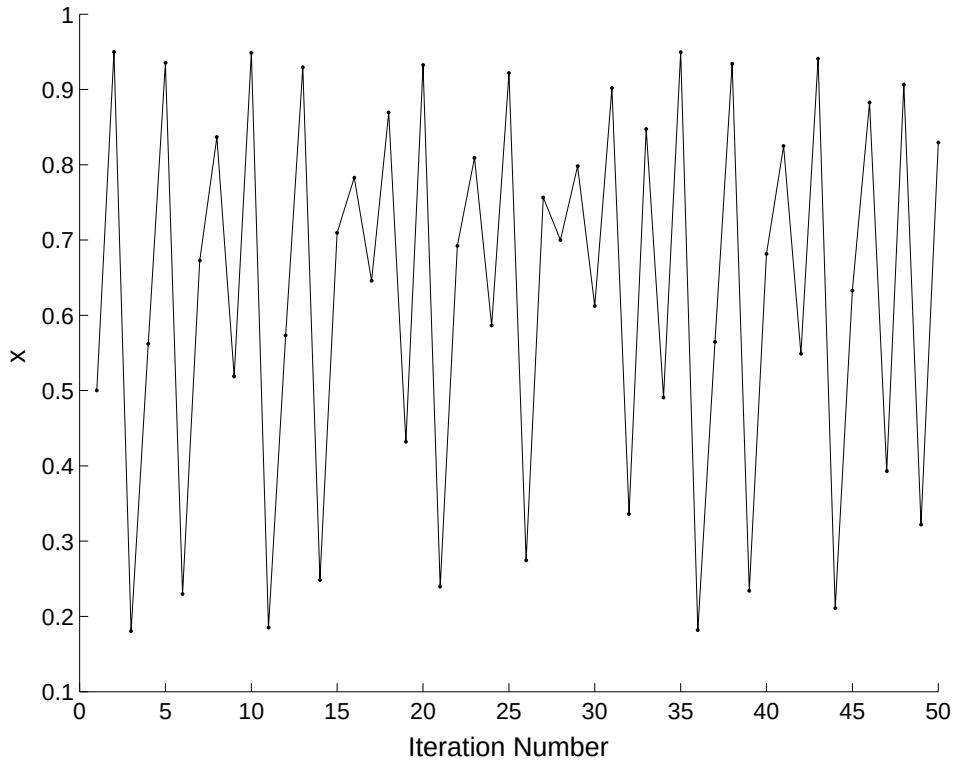

Figure 2: Solution to Eq. (11) with $c = 3.8$.

1.3 Mixing

The fundamental element of the mixing of one fluid within another is called diffusion. This occurs at the atomic level, when molecules, as a result of random molecular motion, intersperse with other molecules [5]. Diffusion can be described as the movement of a species from a region of high concentration to a region of low concentration [6]. To illustrate this point, one could imagine releasing a drop of food coloring in a glass of water. Without any agitation, the colored droplet will diffuse throughout the entire glass, eventually. This type of diffusion can be referred to as ordinary diffusion because it is driven by a concentration gradient. Pressure diffusion and forced diffusion are two other types that are not relevant to this particular work. Diffusion occurs in all circumstances where fluids are in contact, but for diffusion alone to mix two fluids, the time period is often too great for industrial application. Had the imagined glass of water been stirred when the food coloring was added,

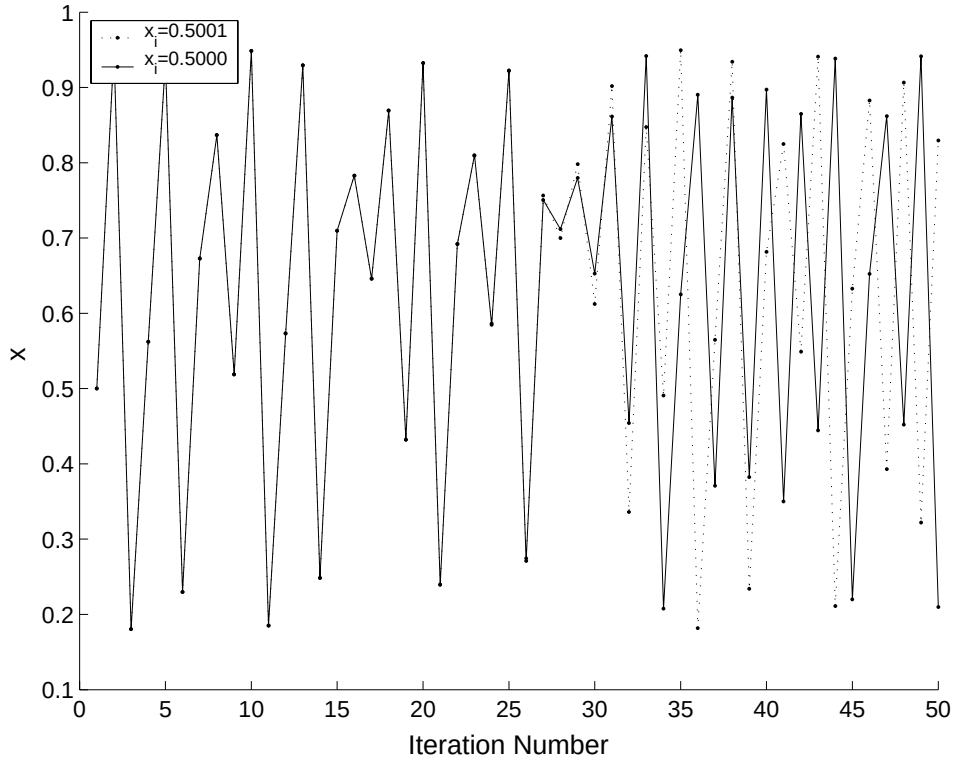

Figure 3: Two solutions to Eq. (11) with $c = 3.8$ and varying initial conditions .

the two fluids would have completely mixed more quickly because the mixing would increase diffusion rates.

Imagining a two fluid system, diffusion occurs over the contact area between the fluids. It follows, then, that to increase the effect of diffusion the surface contact area between the two fluids should be increased. This allows more fluid interspersion to occur, though the rate of that diffusion has not changed. This diffusion of mass is measured in terms of a flux, which is a vector that denotes mass transfer of a species through a unit area per unit time. The flux of fluid A into fluid B in a two fluid system is expressed by Fick's Law, Eq. (12) [6],

$$\boldsymbol{J}_A^\star = -cD_{AB}\nabla x_A, \tag{12}$$

where $\boldsymbol{J}_A^\star$ is the flux of fluid A with units of mass per unit area per unit time, c is the

concentration of fluid A, D_{AB} is a mass diffusivity coefficient with units of length squared over time that varies with each system, and x_A is the mole fraction of fluid A. The mole fraction is defined as the molar concentration of species A divided by the molar density of the entire solution or fluid system, $x_A = c_A/c$. From this relationship, it is easy to see that a mixing system that increases the surface contact area between fluids A and B increases the mass transfer per unit time from one fluid into the other.

Mixing has a special correlation to chaos theory for this particular set of experiments. To achieve the most effective configuration of two vortices for the purpose of mixing, a chaotic system is desirable. A flow field that induces periodic motion will be incapable of producing the increase in surface area that a non-periodic system can create. This is why turbulence, which is chaotic, within a heat exchanger is desirable for efficiency: because the intertwining of the fluid spreads the heat more evenly through the whole of the fluid and thus carries more away. In a similar way, a chaotic two-vortex system will do more to aid diffusion by intertwining the fluids more deeply into each other and increasing surface contact area. A strange attractor, however, may be an exception to this statement. Though a system with a strange attractor is indeed chaotic, there is a possibility that the solution is chaotic within a very bounded area, effectively doing little more than a periodic system.

2 Numerical Methods

2.1 Derivation of Flow Field

To derive the flow field for a two vortex system with variable distance and skew, first a three-dimensional Cartesian coordinate system was chosen to simplify calculations. It appears in Fig. 4. Vortex 1 lies along the Y-axis and does not move or rotate. Vortex 2 lies a distance l along the Z-axis inclined at an angle θ into the page, around the Z-axis, as seen in Fig. 5. The angle θ is measured from the positive Y-axis.

Since velocity terms are additive, the total velocity can be determined by adding the contributions from each vortex. Dealing first with vortex 1, the perpendicular distance r

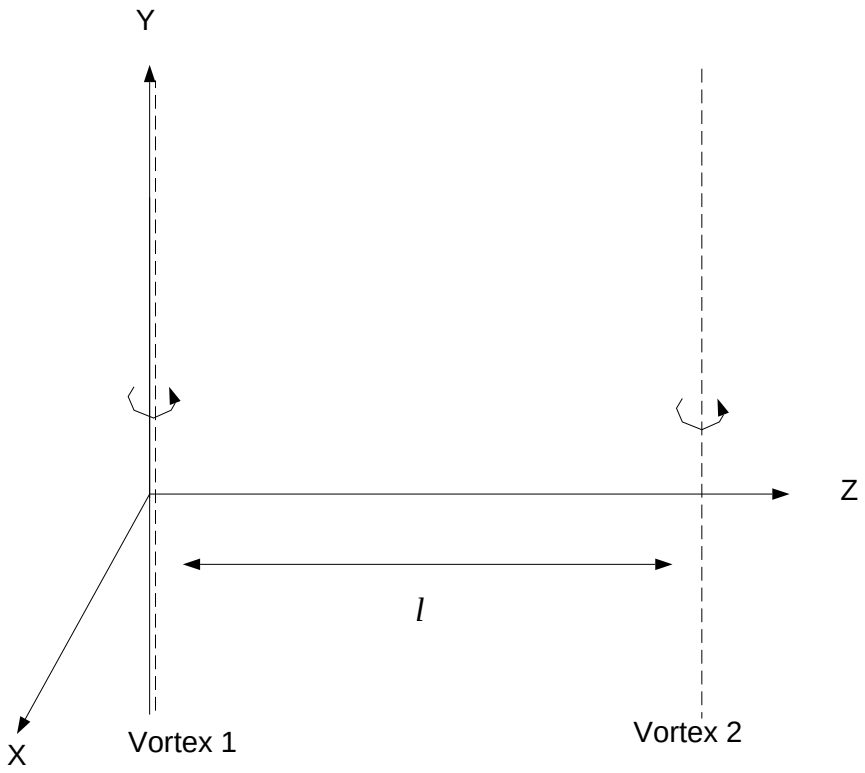

Figure 4: Cartesian coordinate system as applied to two vortices and relevant dimensions

from the line of the vortex to any point in the field is $r = \sqrt{x^2 + z^2}$. Let the angle formed between a line from the vortex to the point and the Z-axis be α. To determine the velocity in the Z-direction, one must multiply the induced angular velocity, given by Eq. (9), by $\sin\alpha$. By inspection, $\sin\alpha = x/\sqrt{x^2 + z^2}$. To find the velocity in the X-direction one must multiply the angular velocity by $z/\sqrt{x^2 + z^2}$. Velocity due to vortex 1 is shown in Eqs. (13), (14) and Eq. (15):

$$\frac{dx}{dt} = \frac{\Gamma}{2\pi}\frac{z}{x^2 + z^2}, \tag{13}$$

$$\frac{dy}{dt} = 0, \tag{14}$$

$$\frac{dz}{dt} = \frac{\Gamma}{2\pi}\frac{x}{x^2 + z^2}. \tag{15}$$

Vortex 1 contributes no velocity in the Y-direction.

To find the velocity induced by vortex 2, a second coordinate system must be imposed

with its origin where vortex 2 intercepts the Z-axis. This second system will rotate with vortex 2 through the angle θ. The axes of this second system are annotated as X_2, Y_2 and Z_2, and they correspond to X, Y and Z, respectively, when $\theta = 0$. A view of the two systems, looking down the Z-axis, is shown in Fig. 5.

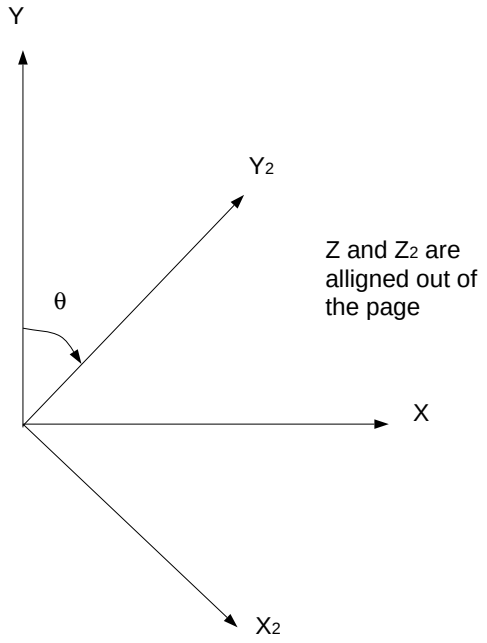

Figure 5: The fixed and rotating coordinate systems as viewed along the Z-axis

Whereas the the fluid particle coordinate can be in terms of X, Y and Z for vortex 1, the coordinates must be mapped into the vortex 2 system before its contributing velocity can be found. This is accomplished using $z_2 = z - l$ and $x_2 = x\cos\theta - y\sin\theta$. Following the same logic as before, the induced velocity of vortex 2 is contained in Eqs. (16), (17)and (18):

$$\frac{dx_2}{dt} = \frac{\Gamma}{2\pi}\frac{z_2}{x_2^2 + z_2^2}, \tag{16}$$

$$\frac{dy_2}{dt} = 0, \tag{17}$$

$$\frac{dz_2}{dt} = \frac{\Gamma}{2\pi}\frac{x_2}{x_2^2 + z_2^2}. \tag{18}$$

To determine combined equations for the flow field, it is first noted that because the Z and Z_2 directions are always aligned, the induced velocities of the vortices in this direction

are simply added together. It is also the case that vortex 2 induces no velocity in the Y_2 direction. Velocity in the X_2 direction, however, corresponds to velocity in both the X and Y direction by the following relationships: $dx/dt = \cos\theta dx_2/dt$ and $dy/dt = -\sin\theta dy_2/dt$. Taking these factors into account and converting all coordinates to measures of X, Y and Z, the flow field is found for the two vector system. It is given in Eq. (19), Eq. (20) and Eq. (21), with initial conditions in Eq. (22):

$$\frac{dx}{dt} = \frac{\Gamma}{2\pi}\left(\frac{z}{x^2+z^2} + \frac{(z-l)\cos\theta}{(x\cos\theta - y\sin\theta)^2 + (z-l)^2}\right), \tag{19}$$

$$\frac{dy}{dt} = -\frac{\Gamma}{2\pi}\frac{(z-l)\sin\theta}{(x\cos\theta - y\sin\theta)^2 + (z-l)^2}, \tag{20}$$

$$\frac{dz}{dt} = -\frac{\Gamma}{2\pi}\left(\frac{x}{x^2+z^2} + \frac{x\cos\theta - y\sin\theta}{(x\cos\theta - y\sin\theta)^2 + (z-l)^2}\right), \tag{21}$$

$$x(0) = L, \qquad y(0) = L, \qquad z(0) = L. \tag{22}$$

2.2 Scaling

The derived flow field contains dimensional parameters and will thus produce dimensional results. To make the results of the examined system more universal, the equations should be scaled. Then, the results are really ratios that can be likened to any other system with any units. To do so, Eqs. (23) and (24) define dimensionless coordinates and a dimensionless parameter, which is a ratio of the distance between the vortices to initial particle position,

$$\tilde{x} = \frac{x}{L}, \qquad \tilde{y} = \frac{y}{L}, \qquad \tilde{z} = \frac{z}{L}, \tag{23}$$

$$\epsilon = \frac{l}{L}. \tag{24}$$

The dimensionless time is given in Eq. (25),

$$\tilde{t} = t\frac{\Gamma}{L^2}, \tag{25}$$

and thus, Eq. (26) follows,

$$\frac{d}{d\tilde{t}} = \frac{L^2}{\Gamma}\frac{d}{dt}. \tag{26}$$

By substituting in Eq. (23), Eq. (25) was found,

$$\frac{d\tilde{x}}{d\tilde{t}} = \frac{L}{\Gamma}\frac{dx}{dt}, \tag{27}$$

with similar results for y and z.

By substituting using Eq. (23) and (24) and multiplying by the term found in Eq. (27), the dimensional flow field is converted to a dimensionless flow field, given in Eq. (28), Eq. (29) and Eq. (30).

$$\frac{d\tilde{x}}{d\tilde{t}} = \frac{1}{2\pi}\left(\frac{\tilde{z}}{\tilde{x}^2+\tilde{z}^2} + \frac{(\tilde{z}-\epsilon)\cos\theta}{(\tilde{x}\cos\theta - \tilde{y}\sin\theta)^2 + (\tilde{z}-\epsilon)^2}\right), \tag{28}$$

$$\frac{d\tilde{y}}{d\tilde{t}} = -\frac{1}{2\pi}\frac{(\tilde{z}-\epsilon)\sin\theta}{(\tilde{x}\cos\theta - \tilde{y}\sin\theta)^2 + (\tilde{z}-\epsilon)^2}, \tag{29}$$

$$\frac{d\tilde{z}}{d\tilde{t}} = -\frac{1}{2\pi}\left(\frac{\tilde{x}}{\tilde{x}^2+\tilde{z}^2} + \frac{\tilde{x}\cos\theta - \tilde{y}\sin\theta}{(\tilde{x}\cos\theta - \tilde{y}\sin\theta)^2 + (\tilde{z}-\epsilon)^2}\right). \tag{30}$$

The initial conditions are scaled to Eq. (31),

$$\tilde{x}(0) = 1, \qquad \tilde{y}(0) = 1, \qquad \tilde{z}(0) = 1. \tag{31}$$

2.3 Numerical Integration

The non-dimensional flow field was solved and iterated numerically using a fourth order Runge-Kutta method. Following this method, if given Eq. (32), the solution is iterated as in Eq. (33) [7].

$$y' = f(t, y) \tag{32}$$

$$y_{n+1} = y_n + h\left(\frac{k_{n1} + 2k_{n2} + 2k_{n3} + k_{n4}}{6}\right) \tag{33}$$

Here, h is a time step that can be assigned any value. The accuracy of the method depends on the value of h. The k functions are given by Eqs. (34) through (37).

$$k_{n1} = f(t_n, y_n) \tag{34}$$

$$k_{n2} = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n1}) \tag{35}$$

$$k_{n3} = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n2}) \tag{36}$$

$$k_{n4} = f(t_n + h, y_n + hk_{n3}) \tag{37}$$

In this case, the flow field is not a function of time, and so just the x, y and z values are substituted for the k functions.

2.4 Error

In numerical integration, the results are subject to two types of error: truncation error and round-off error [7]. Truncation error is a function of the method and is manifested at the level of an individual step giving an overall error in the final value. For the fourth order Runge-Kutta method, the error at each individual step, called local truncation error, has a value of h^5. The truncation error in the final result, called global truncation error, has a value of Ch^4, where C is some constant. Round-off error results from the inability of a computer to carry all the decimal places for the value of a variable. It magnifies as the amount of iterations increases.

These two types of error are dependent upon one another, in that an attempt to prevent one can increase the other. For example, a very small h value will mean a low truncation error both globally and locally, but it will also take more iterations to reach a certain time value. The increased amount of calculations, with the computer losing everything beyond the sixteenth decimal place at each one, increases round-off error. To be effective in numerical calculation, one must choose an h value that minimizes both errors. In the case of the particle paths determined, it was important that output not be subject to error at a large enough scale to appear as movement that does not actually occur. If that were the case, particle movement due to the flow field would be indistinguishable from that caused by numerical error. To solve this problem for the sake of graphical output, a particular time period over which the system was to be iterated was determined, and a system with a known solution was run over this time period. With $\epsilon = \theta = 0$, the two vortices reside above one another and a particle released in the flow field should travel in a circle about the line of the vortices. The time period, seven seconds, was long enough so that the particle would travel around the circle roughly forty times. The value of h was adjusted until there was no graphical derivation around the circle.

Euler's method produced more numerical error, because the local truncation error is is h^2 instead of h^5. The h value had to be so small to guarantee low truncation error that the program required several orders of magnitude more iterations, and round-off error began to accumulate. The higher order of the Runge-Kutta method allows for much more efficient solution.

3 Numerical Simulation Approach

There were two approaches taken to understand the varying effectiveness of vortex configurations. First, a single particle was released in the flow field with initial placement determined by scaling. The parameters ϵ and θ were varied individually to see what type of effect each configuration would have on the motion of a single particle. Single particle motion, however,

was not used to determine the ultimate effectiveness of a system.

To measure effectiveness, a sphere was released within the flow field. The surface of this sphere was defined by 993 points. Each point was allowed to move under the influence of the flow field for a short period of time, allowing the field to slightly deform the surface of the sphere. An algorithm approximated the surface area of the sphere by using Eq. (38) to find the area of the quadrilateral formed by every set of four adjacent points and adding to to find the total.

$$area = \sqrt{(s-a)(s-b)(s-c)(s-d) - abcd\cos^2\left(\frac{(A+B)}{2}\right)} \tag{38}$$

Here, s is found by Eq. (39).

$$s = \frac{a+b+c+d}{2} \tag{39}$$

The geometric equivalent of the variables in Eqs. (38) and (39) can be seen in Fig. 6.

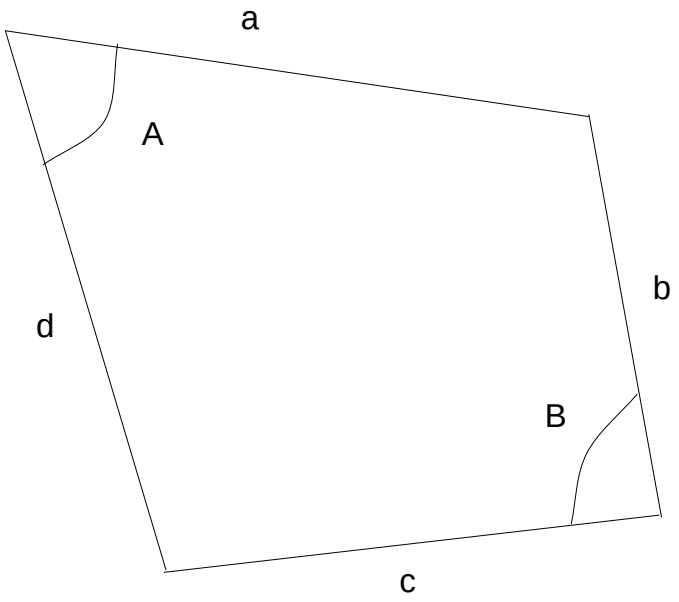

Figure 6: Geometrical representation of variables in Eqs. (38) and (39)

Because it is known that the flow field is incompressible, it is also known that the volume of fluid contained within the sphere will not change. As it is deformed in the flow field,

however, the surface area of the constant volume will change. It is known that the fundamental element of mixing is diffusion, and a way to increase diffusion between fluids that have fixed diffusion parameters is to increase contact surface area. Therefore, it is desirable to maximize the surface area in a minimum time frame. By measuring the surface area before and after the time that the sphere is allowed to deform within the flow field, a dA/dt value can be determined. The system with the highest dA/dt is the most effective because it will produce the most amount of mixing in the shortest amount of time using the same two vortices present in every system, just at different orientation.

Different systems were tested by varying ϵ and θ. Each time the sphere was placed with its center on the XZ-plane, half-way between the two vortices with a constant radius of $\sqrt{3}$.

4 Numerical Simulation Tools

To perform the tasks required, two computer programs were developed. The basic flowcharts for each appear as Figs. 7 and 8.

In the program for a single particle, ϵ and θ can be varied any time the program is run. The initial position of the particle is determined by scaling, as shown in section 2.2. The program iterates the position of the particle by numerically solving Eqs. (28), (29) and (30) using the Runge-Kutta Method. The particle position is written to a file along with Matlab commands, so that after the program is finished, a three dimensional graph of the particle path can be determined. A print step is chosen beforehand, which determines how many iterations will run between the times when the particle position is printed to the output file. The number can be changed to satisfy the desired resolution of the graph.

The algorithm that deforms a sphere in the flow field allows the choice of the same parameters. The radius is fixed, because it must also satisfy the scaled initial conditions in that a point L distance in the X, Y and Z directions must reside on the surface of the sphere. This makes the radius a constant of $\sqrt{3}$. The points on the surface of the sphere are defined using polar coordinates. A full longitudinal ring of points is defined by iterating ϕ,

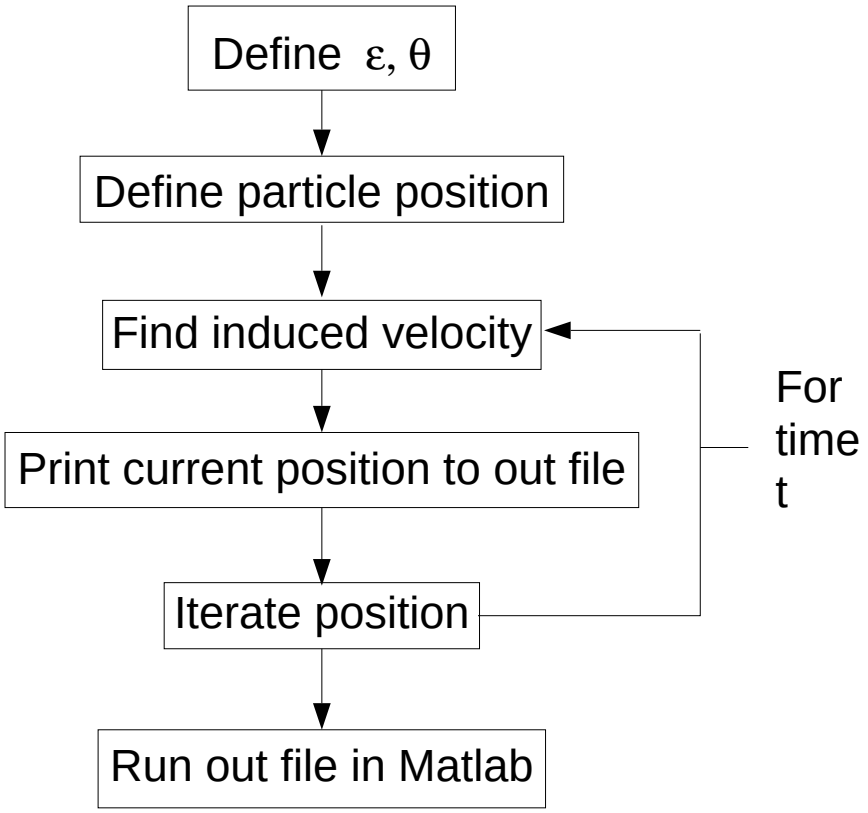

Figure 7: Flowchart showing the method of a program for finding the path of a single particle through the flow field

and after each ring is completed, θ of the coordinate system is iterated through a distance of π. After each iteration, the point is mapped into the Cartesian system so it can be run in the existing equations for the flow field. Each particle is displaced in the same way as in the single particle program. The time of the simulation, however, is much shorter so that the sphere only slightly deforms. The surface area is approximated using Eqs. (38) and (39), and $d\tilde{A}/d\tilde{t}$ is calculated by finding the change in surface area and dividing by the scaled time. The initial and final position of each point is written to an output file, which creates a three-dimensional graphical comparison between the two spheres.

5 Results

To show that the flow field was modeled correctly, a test can be performed by running the single particle program with a vortex configuration of known effect. When the distance l between the vortices is zero and the angle θ is zero, the vortices reside above one another and should cause the particle to travel in a circle with its center at the vortices. Fig. 9 shows the output for this system, which performed as was expected. This system can also be used to estimate the error in the applied numerical methods. Fig. 18 shows the accumulated error in the distance r from the line vortex in the circular path system from Fig. 9. By dividing the value of r for each dimensionless timestep by the initial value of r, the percent error in the position of th particle is determined. The plot goes through $\tilde{t} = 4200$, which is the length of dimensionless time for which all the single particle systems were run.

It was found that when the two vortices intersect, regardless of the value of θ, the particle path is a closed loop. This is demonstrated in Fig. 10. When the flow field is examined in light of the constraints $\theta = \pi/2$ and $\epsilon = 0$, Eqs. (28), (29) and (30) reduce to Eqs. (40), (41) and (42), explaining this occurance.

$$\frac{d\tilde{x}}{d\tilde{t}} = \frac{1}{2\pi}\frac{\tilde{z}}{\tilde{x}^2 + \tilde{z}^2} \tag{40}$$

$$\frac{d\tilde{y}}{d\tilde{t}} = -\frac{1}{2\pi}\frac{\tilde{z}}{\tilde{y}^2 + \tilde{z}^2} \tag{41}$$

$$\frac{d\tilde{z}}{d\tilde{t}} = -\frac{1}{2\pi}\frac{\tilde{x}}{\tilde{x}^2 + \tilde{z}^2} + \frac{1}{2\pi}\frac{\tilde{y}}{\tilde{y}^2 + \tilde{z}^2} \tag{42}$$

Eqs. (43) and (44) follow.

$$\tilde{x}\frac{d\tilde{x}}{d\tilde{t}} + \tilde{y}\frac{d\tilde{y}}{d\tilde{t}} + \tilde{z}\frac{d\tilde{z}}{d\tilde{t}} = 0 \tag{43}$$

$$\int \tilde{x}d\tilde{x} + \int \tilde{y}d\tilde{y} + \int \tilde{z}d\tilde{z} = 0 \tag{44}$$

Solving Eq. (44) gives Eq. (45), the equation for a sphere. When all the coordinates of the plot in Fig. 10 are substituted into Eq. (45), they do all equal the same constant, save a slight variation due to numerical error. This holds true for any angle θ as long as $\epsilon = 0$.

$$\tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2 = C \tag{45}$$

Figs. 11 through 14 show a particle released in a chaotic system. In each, ϵ is fixed and θ is varied. The particle path will not repeat. For the two vortex system to become non-periodic, ϵ must not equal zero. If this is true, then the system is chaotic regardless of the angle θ.

Figs. 15 through 17 also show a series of chaotic systems with fixed θ and varying ϵ. The figure for $\epsilon = 2$ can already be seen as Fig. 13.

Fig. 19 shows a graph generated by the sphere deformation program. The longitudinal lines show the surface of the sphere as it was originally defined. The dots show the same points defining the surface after a period of deformation.

Since it was already known from single particle investigation that intersecting vortices would not increase the surface area of the sphere if they intersected at its origin, testing began at $\epsilon = 1$ and stopped at $\epsilon = 4$, where it was clear that the $d\tilde{A}/d\tilde{t}$ value had already peaked. These results were condensed into a single graph, Fig. 20, showing $d\tilde{A}/d\tilde{t}$ versus ϵ for varying θ.

6 Discussion

The limited case was studied, that two ideal vortices of equal strength are needed to quickly and thoroughly mix a sphere of fluid A, situated between the vortices, into surrounding fluid B. It was found that the most effective vortex configuration is to have a distance between the vortices equal to the diameter of the sphere with an angle between the vortices of $\pi/4\ rad$. These results make sense physically. It is known that if the two vortices intersect, at any angle, at the origin of the sphere, the rate of change of surface area will be zero. Every

particle will be confined to a path on the surface of the fluid sphere. As the vortices are moved apart, keeping the origin of the sphere between them, the rate of change of surface area increases. If the vortices continue to move apart so that they reside far beyond the boundaries of the sphere, then there will be little induced velocity because of the distance from the lines of the vortices. Therefore, there is a peak in the attainable change in surface area between these two limits. Physically, it is easy to imagine it might reside at the only physical boundary between these limits: the point where the vortices leave the volume of the sphere. In the computer simulations, it was not possible to position the vortices right at the boundary of the sphere, because this would have left the points at the top and bottom of the sphere on the line of the vortex, creating a singularity. The distance used, to approximate the change in surface area at the diameter, 3.5, is only slightly larger than the actual diameter of the sphere, and so it gives a fair approximation.

Calculating a good approximation of the deformed sphere is crucial to quantifying the effectiveness of the compared systems. Because all spheres had a radius of $\sqrt{3}$, it is possible to calculate the correct initial surface area for any one of them as $A = 4\pi r^2 = 12\pi = 37.7$. The computer simulations also found this initial area to compare to the final area for calculating $d\tilde{A}/d\tilde{t}$. Values of this calculation ranged from 37.9 to 38.17. Taking the largest error in surface area, it is 1.25 percent of the actual A. The reason the computer program used a calculated initial area value was because it was the approximated figure of a sphere and not an actual sphere that was being deformed in the flow field. The degree of resolution in the surface that contributed to the initial overestimate was likely to work the same way in finding the final volume. Comparing like solutions treats these imperfections rather as a part of the geometry. Using the theoretical area result would have magnified errors, even though it more correctly defined the shape being approximated.

It is interesting to imagine that each of the points on the deformed sphere is traveling in a chaotic path like some of the single particle figures contained here. Other than saying that the introduction of an ϵ value adds new terms to the flow field that no longer allow it to

reduce to the equation for a sphere, it is not yet determined what exactly is changing in the system to lead to such disorder from such order. This an an area where the project should be furthered: obtaining a mathematical understanding of the transition into chaos.

Other areas where the project could be furthered include numerical simulation of the same systems but with unequal vortex strength. It would also be interesting to release several particles near each other in the flow field, plotting the path of each on the same graph and determining how much the paths diverge. More accurate results for the rate of change in surface area could be obtained by refining the points on the surface of the released sphere.

Although the system considered here is ideal, the results have real world application. If a stream of fluid A was released into fluid B, it should most likely enter the fluid between two vortices oriented in the manner found here, with the distance between equal to the cross-sectional diameter of the stream. This is an idea applicable to all types of entrance flows or to scenarios in industry where vats of fluid at rest need to be mixed. Energy can be saved if the vortices are oriented properly to maximize the effect of the power driving the system. If vortices are induced within internal pipe flows such as mufflers and heat exchangers, it would be of benefit to design the anomalies along the length of the pipes to create vortices of this orientation, as increased effectiveness would require less to be made.

References

[1] Fox, R. W., and McDonald, A. T., 1998, *Introduction to Fluid Mechanics*, John Wiley and Sons, Inc., New York.

[2] Shepherd, D. G., 1965, *Elements of Fluid Mechanics*, Harcourt, Brace & World, Inc., New York.

[3] Potter, M. C., and Wiggert, 1991, *Mechanics of Fluids*, Prentice Hall, Englewood Cliffs, NJ.

[4] Gleick, J., 1987, *Chaos: Making a New Science*, Penguin Books, New York.

[5] Cussler, E. L., 1984, *Diffusion: Mass Transfer in Fluid Systems*, Cambridge University Press, Cambridge, England.

[6] Bird, R. B., Stewart, and Lightfoot, 1960, *Transport Phenomena*, John Wiley and Sons, Inc., New York.

[7] Boyce, W. E., and DiPrima, 2001, *Elementary Differential Equations and Boundary Value Problems*, Seventh Edition, John Wiley and Sons, Inc., New York.

[8] Powers, Joseph M., Class Notes, Fluid Mechanics.

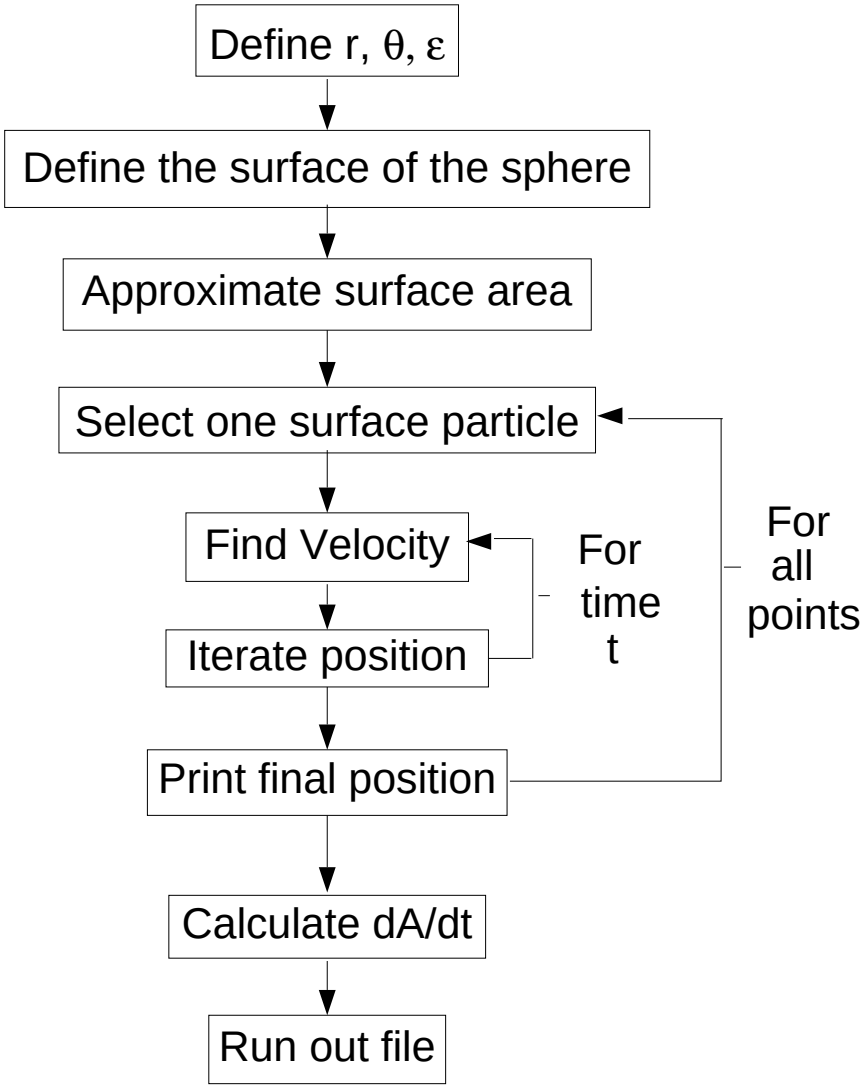

Figure 8: Flowchart showing the method of a program for finding the change in area of a sphere released in the flow field

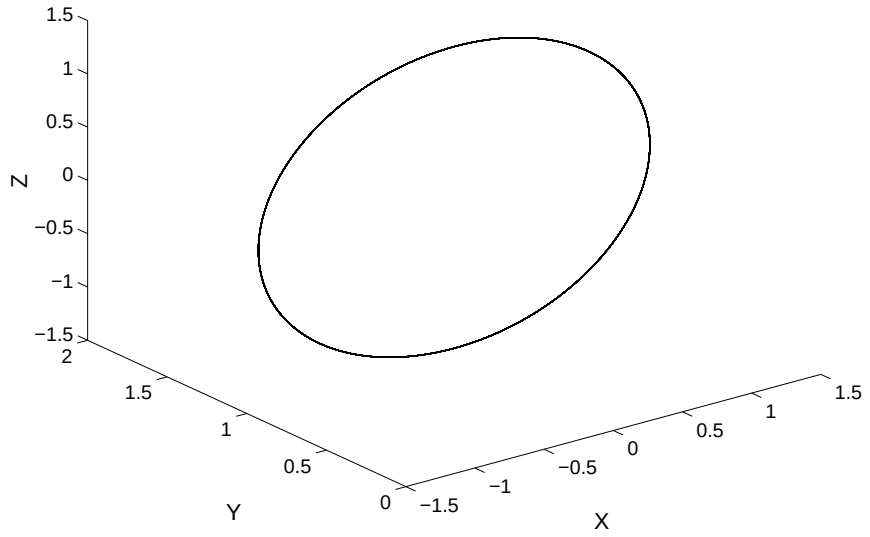

Figure 9: The particle travels in a circle around the vortices when ϵ and θ are zero. There is no change of position in the Y-direction.

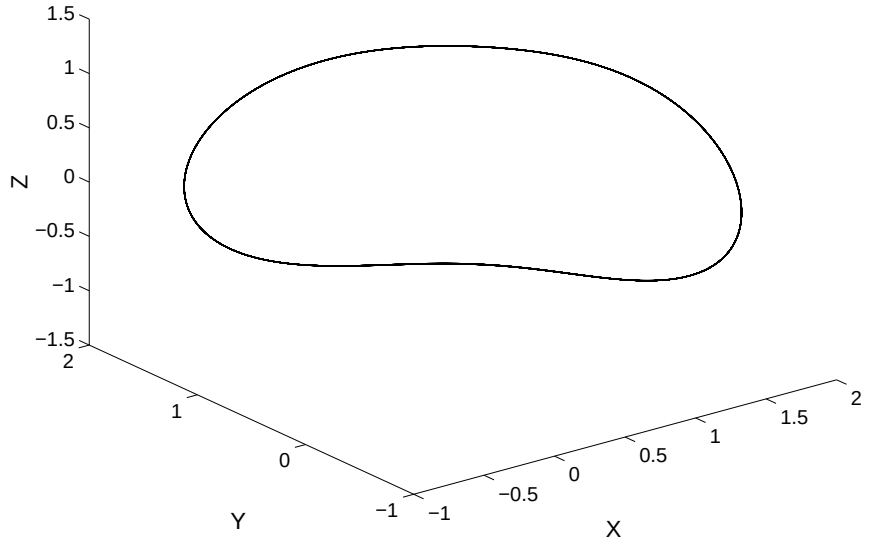

Figure 10: The particle path is periodic on the surface of a sphere for all θ when ϵ is zero. Here, $\theta = \pi/2$.

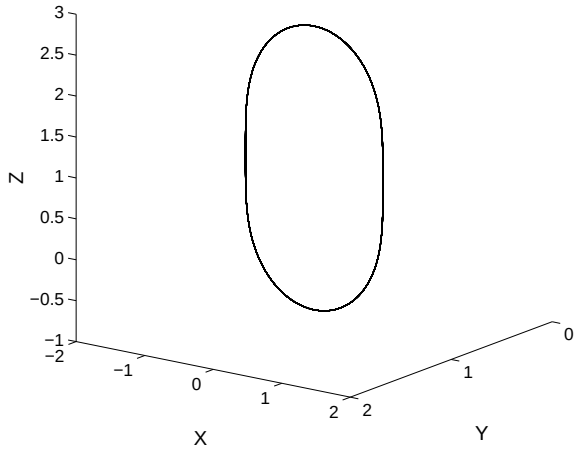

Figure 11: A three dimensional view. Here, $\epsilon = 2$, $\theta = 0$.

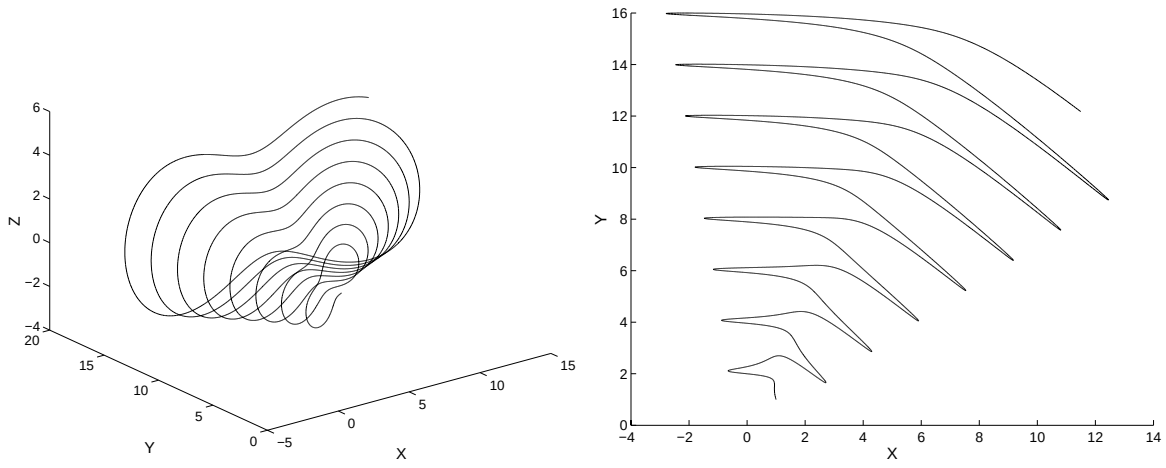

Figure 12: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 2$, $\theta = \pi/4$.

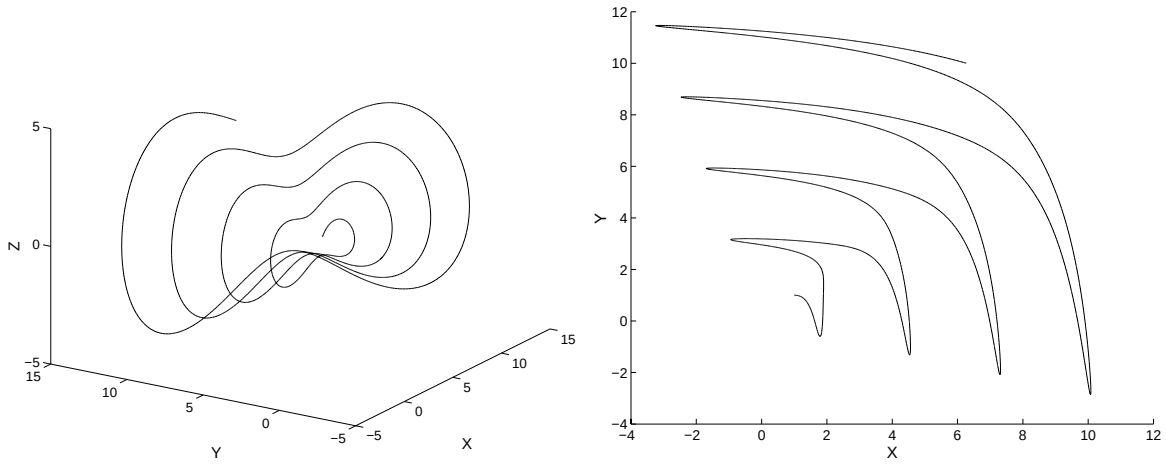

Figure 13: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 2$, $\theta = \pi/2$.

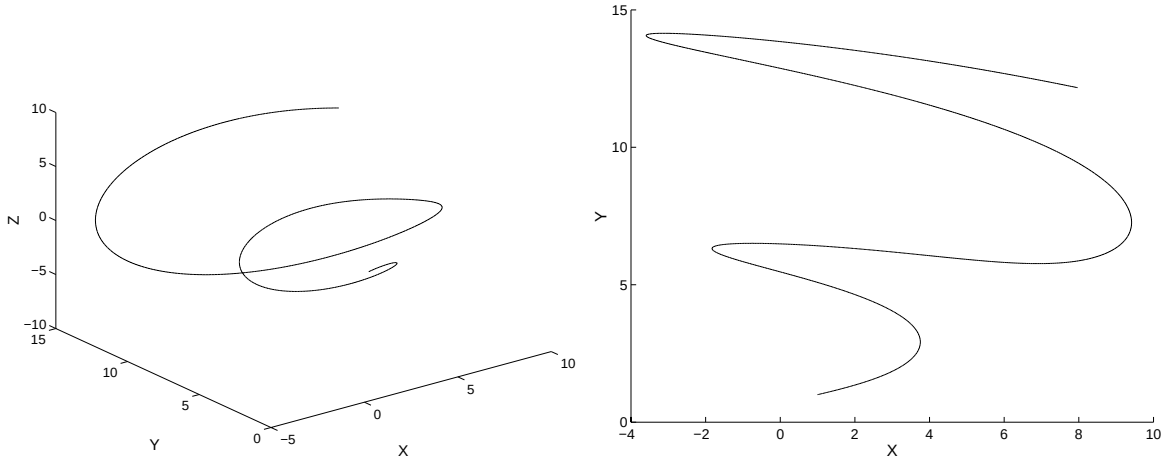

Figure 14: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 2$, $\theta = 3\pi/2$.

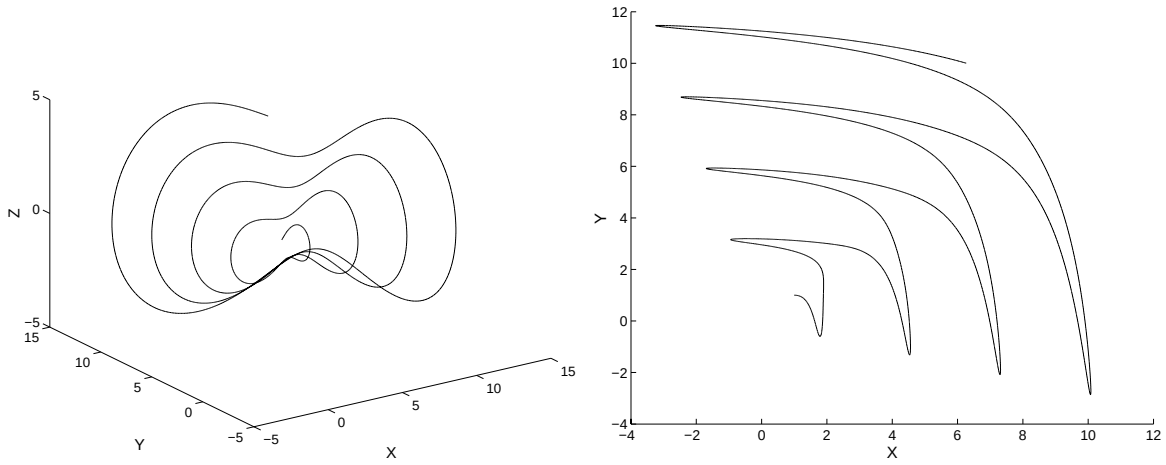

Figure 15: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 1$, $\theta = \pi/2$.

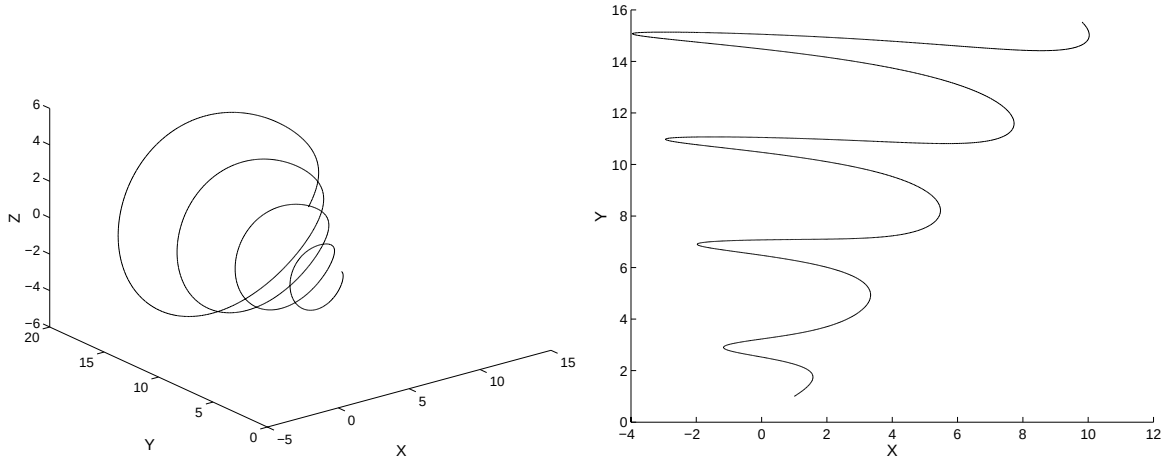

Figure 16: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 3$, $\theta = \pi/2$.

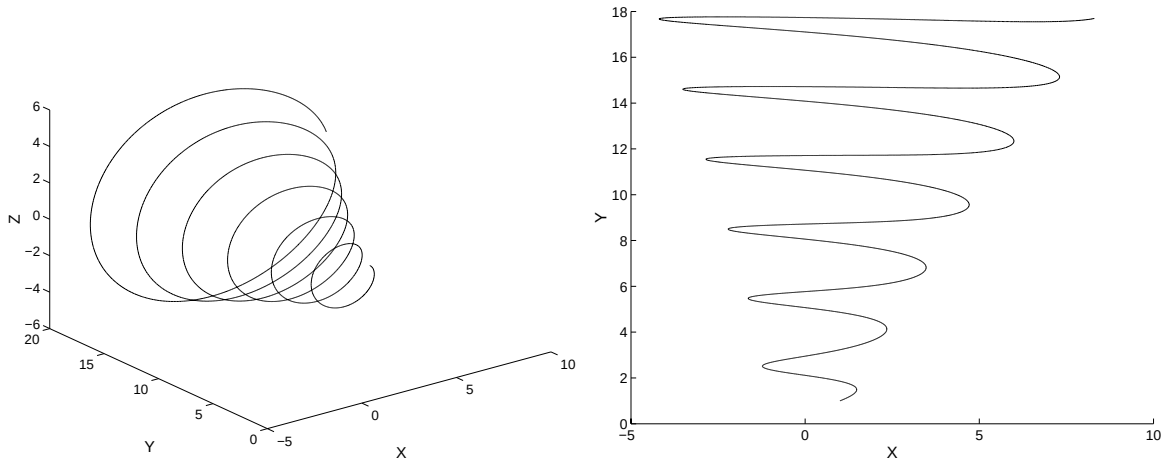

Figure 17: A three dimensional view and projection to the XY-plane. Here, $\epsilon = 4$, $\theta = \pi/2$.

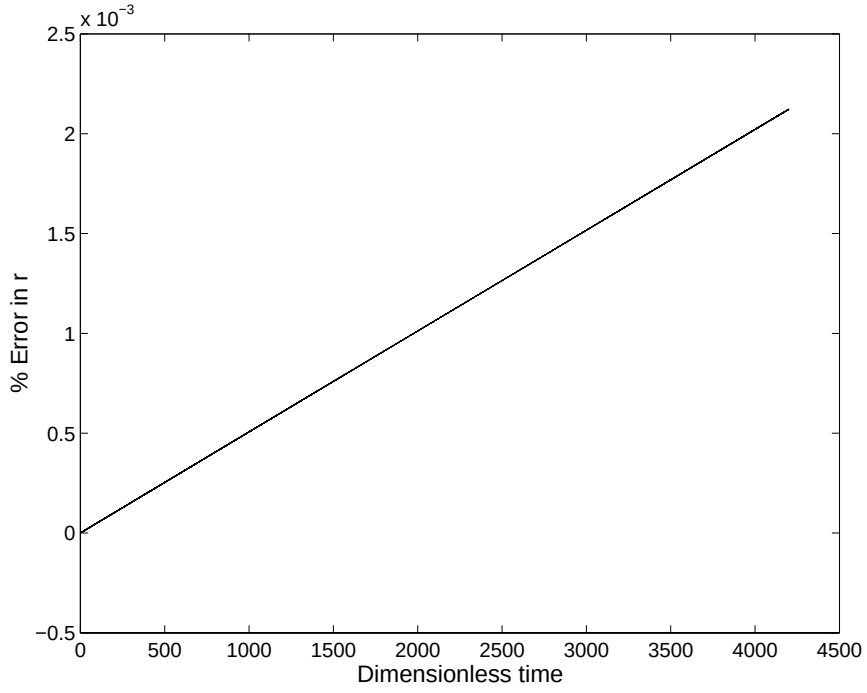

Figure 18: Percent error in r for $\epsilon = \theta = 0$ *vs.* $\tilde{t}$.

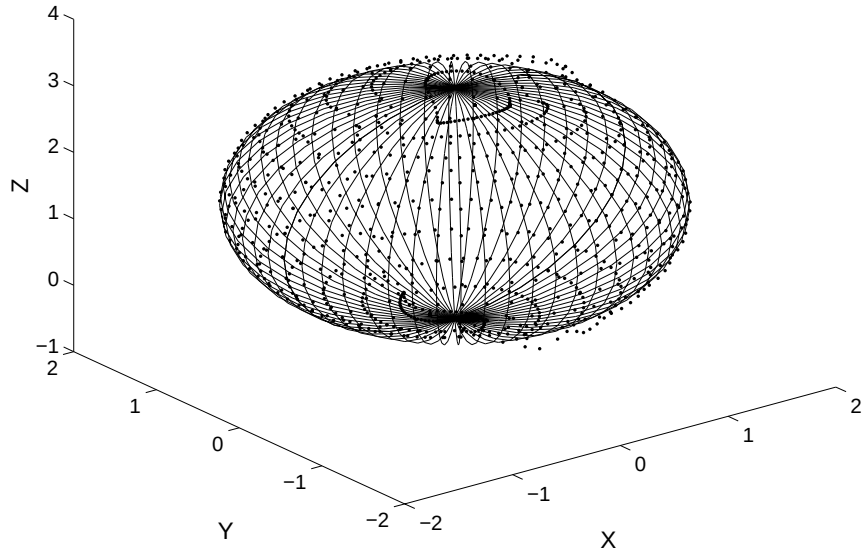

Figure 19: The initial (lines) and later deformed (points) surface of the sphere. Here, $\epsilon = 3$ and $\theta = \pi/2$.

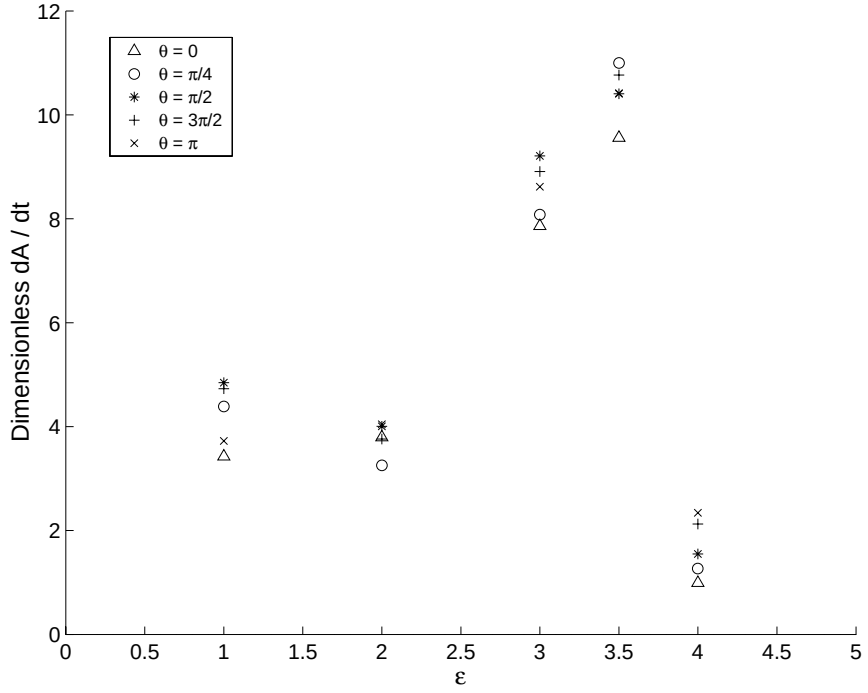

Figure 20: $d\tilde{A}/d\tilde{t}$ versus ϵ for varying θ.

```
/* A program to simulate the three dimensional motion of a fluid particle with velocity induced by two skewed vortices */
#include <stdio.h> #include <stdlib.h> #include <math.h>
main(){
/*initialize variables*/
/*Distance between vortices*/ double e; /*Skiew between vortices*/ double theta; /* Coordinates of the fluid particle */ double Ax, Ay, Az;
/*Runge-Kutta functions*/ double k1x, k2x, k3x, k4x, k1y, k2y, k3y, k4y, k1z, k2z, k3z, k4z; /*Print Step*/ int print; /*Loop counter*/ int i;
/*time step*/ double h=6e-5; /*PI*/ double c=22.0/7.0; /*Run time*/ double t; /*Initial point length from all axes*/ double L;
/*file output*/ FILE *fp;fp = fopen("output_nd1.m", "w");
/*user input*/
printf("Seperation to initial point ratio:\n"); scanf("%lf",&e);
printf("Angle Between Vortices:\n"); scanf("%lf",&theta);
Ax=1; Ay=1; Az=1;
printf("Run Time:\n"); scanf("%lf",&t);
printf("Print step:\n"); scanf("%i",&print);
/*iterate particle position*/
fprintf(fp, "a=[");
for(i=0;i<=t/h; i++){
        if(i%print==0){ fprintf(fp, "%lf, %lf, %lf;\n",Ax, Ay, Az);}
        k1z=-(1/(2.0*c))*(Ax/(Ax*Ax+Az*Az))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k2z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+0.5*h*k1z)*(Az+0.5*h*k1z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+0.5*h*k1z)-e)*((Az+0.5*h*k1z)-e));
        k3z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+0.5*h*k2z)*(Az+0.5*h*k2z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+0.5*h*k2z)-e)*((Az+0.5*h*k2z)-e));
        k4z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+h*k3z)*(Az+h*k3z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+h*k3z)-e)*((Az+h*k3z)-e));
        k1x=(1/(2.0*c))*(Az/(Ax*Ax+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k2x=(1/(2.0*c))*(Az/((Ax+0.5*h*k1x)*(Ax+0.5*h*k1x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                (((Ax+0.5*h*k1x)*cos(theta)-Ay*sin(theta))*((Ax+0.5*h*k1x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k3x=(1/(2.0*c))*(Az/((Ax+0.5*h*k2x)*(Ax+0.5*h*k2x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                (((Ax+0.5*h*k2x)*cos(theta)-Ay*sin(theta))*((Ax+0.5*h*k2x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k4x=(1/(2.0*c))*(Az/((Ax+h*k3x)*(Ax+h*k3x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                (((Ax+h*k3x)*cos(theta)-Ay*sin(theta))*((Ax+h*k3x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k1y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
        k2y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+0.5*h*k1y)*sin(theta))*
                (Ax*cos(theta)-(Ay+0.5*h*k1y)*sin(theta))+(Az-e)*(Az-e));
        k3y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+0.5*h*k2y)*sin(theta))*
                (Ax*cos(theta)-(Ay+0.5*h*k2y)*sin(theta))+(Az-e)*(Az-e));
        k4y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+h*k3y)*sin(theta))*
                (Ax*cos(theta)-(Ay+h*k3y)*sin(theta))+(Az-e)*(Az-e));
        Ax=Ax+h*((k1x+2*k2x+2*k3x+k4x)/6);   Ay=Ay+h*((k1y+2*k2y+2*k3y+k4y)/6);   Az=Az+h*((k1z+2*k2z+2*k3z+k4z)/6);}
fprintf(fp, "]; \n"); fprintf(fp, "plot3(a(:,1),a(:,2),a(:,3))");}
```

Figure 21: C code numerically simulating single particle motion

```
/* A program to simulate the deformation of a fluid sphere in a flow field induced by two skewed and seperated vortices */
#include <stdio.h> #include <stdlib.h> #include <math.h>
main(){
/*Angle between vortices*/ double theta; /*Ratio of skiew between vortices to initial position*/ double e; /*Coordinates of the fluid particle */ double Ax, Ay, Az;
/*Runge-Kutta functions*/ double k1x, k2x, k3x, k4x, k1y, k2y, k3y, k4y, k1z, k2z, k3z, k4z, dxdt, dydt, dzdt; /*Loop counters*/ int i, k; /*time step*/double h=6e-5;
/*PI*/ double c=22.0/7.0; /*Run time*/ double t; /*Sphere determining angles*/ double phi=0.0, psi=0.0; /*sphere*/ double spherex[993], spherey[993], spherez[993];
/*sphere center and radius*/ double sx, sy, sz, r; /*Area calculation*/ double a, b, cc, d, s, A, C, initial=0.0, area=0.0;
/*file output*/ FILE *fp;fp = fopen("spherend1.m", "w");
/*user imput*/
printf("e:\n"); scanf("%lf",&e);
printf("Angle Between Vortices:\n"); scanf("%lf",&theta);
printf("sphere center coordinates; x:\n"); scanf("%lf",&sx);
printf("sphere center coordinates; y:\n"); scanf("%lf",&sy);
printf("sphere center coordinates; z:\n"); scanf("%lf",&sz);
printf("sphere radius; r:\n"); scanf("%lf",&r);
printf("Run Time:\n"); scanf("%lf",&t);
/*define sphere surface*/
for(i=0; i<=992; i++){
        spherex[i]=sx+r*sin(phi)*cos(psi); spherey[i]=sy+r*sin(phi)*sin(psi); spherez[i]=sz+r*cos(phi);
        if(i%32==0){psi=psi+0.104720;}
        phi=phi+0.196349;}
/*print surface to out file for plotting*/
fprintf(fp, "a=[");
for(i=0; i<=992; i++){fprintf(fp, "%lf, %lf, %lf;\n",spherex[i], spherey[i], spherez[i]);}
fprintf(fp, "]; \n"); fprintf(fp, "plot3(a(:,1),a(:,2),a(:,3))\n\n"); fprintf(fp, "hold on \n b=[");
/*calculate initial area*/
for(i=0;i<=960;i++){
        a=sqrt((spherex[i+1]-spherex[i])*(spherex[i+1]-spherex[i])+ (spherey[i+1]-spherey[i])*(spherey[i+1]-spherey[i])+(spherez[i+1]-spherez[i])*(spherez[i+1]-spherez[i]));
        b=sqrt((spherex[i+32]-spherex[i])*(spherex[i+32]-spherex[i])+(spherey[i+32]-spherey[i])*(spherey[i+32]-spherey[i])+(spherez[i+32]-spherez[i])*(spherez[i+32]-spherez[i]));
        cc=sqrt((spherex[i+32]-spherex[i+33])*(spherex[i+32]-spherex[i+33]) +(spherey[i+32]-spherey[i+33])*
                (spherey[i+32]-spherey[i+33])+(spherez[i+32]-spherez[i+33])*(spherez[i+32]-spherez[i+33]));
        d=sqrt((spherex[i+1]-spherex[i+33])*(spherex[i+1]-spherex[i+33])+(spherey[i+1]-spherey[i+33])*(spherey[i+1]-spherey[i+33])+(spherez[i+1]-spherez[i+33])*
                (spherez[i+1]-spherez[i+33]));
        s=(a+b+cc+d)/2;
        A=acos((abs(spherex[i+1]-spherex[i])*abs(spherex[i+1]-spherex[i+33])+abs(spherey[i+1]-spherey[i])*
                abs(spherey[i+1]-spherey[i+33])+abs(spherez[i+1]-spherez[i])*abs(spherez[i+1]-spherez[i+33]))/(a*d));
        C=acos((abs(spherex[i+32]-spherex[i])*abs(spherex[i+32]-spherex[i+33])+abs(spherey[i+32]-spherey[i])*
                abs(spherey[i+32]-spherey[i+33])+abs(spherez[i+32]-spherez[i])*abs(spherez[i+32]-spherez[i+33]))/(b*cc));
        if(a<=0.0001){a=0.0;}if(b<=0.0001){b=0.0;}if(c<=0.0001){c=0.0;}if(d<=0.0001){d=0.0;}
        initial=initial+sqrt((s-a)*(s-b)*(s-cc)*(s-d)-a*b*cc*d*cos(0.5*(A+C))*cos(0.5*(A+C)));}
```

Figure 22: C code numerically simulating sphere deformation and finding change in area (continued next page)

```
/*iterate each particle in the flow field*/
for(i=0;i<=992; i++){
        for(k=0;k<=t/h; k++){
                Ax=spherex[i]; Ay=spherey[i]; Az=spherez[i];
                k1z=-(1/(2.0*c))*(Ax/(Ax*Ax+Az*Az))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/((Ax*cos(theta)-Ay*sin(theta))*
                        (Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k2z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+0.5*h*k1z)*(Az+0.5*h*k1z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                        ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+0.5*h*k1z)-e)*((Az+0.5*h*k1z)-e));
                k3z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+0.5*h*k2z)*(Az+0.5*h*k2z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                        ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+0.5*h*k2z)-e)*((Az+0.5*h*k2z)-e));
                k4z=-(1/(2.0*c))*(Ax/(Ax*Ax+(Az+h*k3z)*(Az+h*k3z)))-(1/(2.0*c))*(Ax*cos(theta)-Ay*sin(theta))/
                        ((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+((Az+h*k3z)-e)*((Az+h*k3z)-e));
                k1x=(1/(2.0*c))*(Az/(Ax*Ax+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/((Ax*cos(theta)-Ay*sin(theta))*
                        (Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k2x=(1/(2.0*c))*(Az/((Ax+0.5*h*k1x)*(Ax+0.5*h*k1x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                        (((Ax+0.5*h*k1x)*cos(theta)-Ay*sin(theta))*((Ax+0.5*h*k1x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k3x=(1/(2.0*c))*(Az/((Ax+0.5*h*k2x)*(Ax+0.5*h*k2x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                        (((Ax+0.5*h*k2x)*cos(theta)-Ay*sin(theta))*((Ax+0.5*h*k2x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k4x=(1/(2.0*c))*(Az/((Ax+h*k3x)*(Ax+h*k3x)+Az*Az))+(1/(2.0*c))*((Az-e)*cos(theta))/
                        (((Ax+h*k3x)*cos(theta)-Ay*sin(theta))*((Ax+h*k3x)*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k1y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-Ay*sin(theta))*(Ax*cos(theta)-Ay*sin(theta))+(Az-e)*(Az-e));
                k2y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+0.5*h*k1y)*sin(theta))*
                        (Ax*cos(theta)-(Ay+0.5*h*k1y)*sin(theta))+(Az-e)*(Az-e));
                k3y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+0.5*h*k2y)*sin(theta))*
                        (Ax*cos(theta)-(Ay+0.5*h*k2y)*sin(theta))+(Az-e)*(Az-e));
                k4y=-(1/(2.0*c))*((Az-e)*sin(theta))/((Ax*cos(theta)-(Ay+h*k3y)*sin(theta))*
                        (Ax*cos(theta)-(Ay+h*k3y)*sin(theta))+(Az-e)*(Az-e));
                spherex[i]=Ax+h*((k1x+2*k2x+2*k3x+k4x)/6);
                spherey[i]=Ay+h*((k1y+2*k2y+2*k3y+k4y)/6);
                spherez[i]=Az+h*((k1z+2*k2z+2*k3z+k4z)/6);}
        fprintf(fp, "%lf, %lf, %lf;\n",spherex[i], spherey[i], spherez[i]);}
fprintf(fp, "]; \n");
fprintf(fp, "plot3(b(:,1),b(:,2),b(:,3))\n\n");
```

Figure 23: C code numerically simulating sphere deformation and finding change in area (continued next page)

```
/*calculate final area*/
for(i=0;i<=960;i++){
        a=sqrt((spherex[i+1]-spherex[i])*(spherex[i+1]-spherex[i])+ (spherey[i+1]-spherey[i])*
                (spherey[i+1]-spherey[i])+(spherez[i+1]-spherez[i])*(spherez[i+1]-spherez[i]));
        b=sqrt((spherex[i+32]-spherex[i])*(spherex[i+32]-spherex[i])+(spherey[i+32]-spherey[i])*
                (spherey[i+32]-spherey[i])+(spherez[i+32]-spherez[i])*(spherez[i+32]-spherez[i]));
        cc=sqrt((spherex[i+32]-spherex[i+33])*(spherex[i+32]-spherex[i+33]) +(spherey[i+32]-spherey[i+33])*
                (spherey[i+32]-spherey[i+33])+(spherez[i+32]-spherez[i+33])*(spherez[i+32]-spherez[i+33]));
        d=sqrt((spherex[i+1]-spherex[i+33])*(spherex[i+1]-spherex[i+33])+(spherey[i+1]-spherey[i+33])*
                (spherey[i+1]-spherey[i+33])+(spherez[i+1]-spherez[i+33])*(spherez[i+1]-spherez[i+33]));
        s=(a+b+cc+d)/2;
        A=acos((abs(spherex[i+1]-spherex[i])*abs(spherex[i+1]-spherex[i+33])+abs(spherey[i+1]-spherey[i])*
                abs(spherey[i+1]-spherey[i+33])+abs(spherez[i+1]-spherez[i])*abs(spherez[i+1]-spherez[i+33]))/(a*d));
        C=acos((abs(spherex[i+32]-spherex[i])*abs(spherex[i+32]-spherex[i+33])+abs(spherey[i+32]-spherey[i])*
                abs(spherey[i+32]-spherey[i+33])+abs(spherez[i+32]-spherez[i])*abs(spherez[i+32]-spherez[i+33]))/(b*cc));
        if(a<=0.0001){a=0.0;}if(b<=0.0001){b=0.0;}if(c<=0.0001){c=0.0;}if(d<=0.0001){d=0.0;}
        area=area+sqrt((s-a)*(s-b)*(s-cc)*(s-d)-a*b*cc*d*cos(0.5*(A+C))*cos(0.5*(A+C)));}
printf("Initial area of sphere:\n");
printf("%lf\n",initial);
printf("Final area of sphere:\n");
printf("%lf\n",area);
printf("dA/dt:\n");
printf("%lf\n",(area-initial)/t);}
```

Figure 24: C code numerically simulating sphere deformation and finding change in area