

A Numerical Study of the Swirling Vortex

Reza Malek-Madani
Department of Mathematics
U.S. Naval Academy
Annapolis, MD 21402

James E. Coleman
Department of Oceanography
U. S. Naval Academy
Annapolis, MD

David R. Smith
Department of Oceanography
U. S. Naval Academy
Annapolis, MD 21412

December 4, 2003

Abstract

In this paper we present the implementation of a numerical algorithm to simulate the swirling vortex, a particular solution of the Navier-Stokes equations. All of the computations presented, whether numerical or symbolic, are carried out in *Mathematica* where the code is displayed in its entirety. Applying a fixed-point iteration scheme in combination with the Galerkin method, we obtain an approximate solution to a boundary-value problem for the underlying nonlinear system of integro-differential equations. The solution of this system represents the velocity field for the motion of a viscous fluid induced by a line vortex and bounded by a plane boundary surface perpendicular to the vortex line. We next consider the Lagrangian dynamical system that results from this Eulerian velocity field and simulate the motion of parcels of fluid under the action of the vortex line for various values of the parameters associated with fluid viscosity and vortex strength.

1 Introduction

The numerical study in this paper is motivated by the analysis presented in [1], by J. Serrin, and in [2], by M. A. Goldshtik, in which a particular steady-state solution to the Navier-Stokes equations is studied. This solution describes a viscous flow in half-space induced by a line vortex in which fluid particles are allowed to adhere to the bounding surface thus simulating the idealized dynamic flow of an atmospheric tornado in contact with the ground. As a model for tornadic flow, this solution has received

attention recently in [3] and [4] where its physical significance relative to other models are discussed in some detail.

As we will show in the next section, the analysis of this steady-state flow is reduced to the study of a boundary-value problem for the following system of integro-differential equations:

$$\begin{aligned} f' + f^2 &= k^2 \frac{Q(x)}{(1-x^2)^2} \\ \Omega'' + 2f\Omega' &= 0, \end{aligned} \quad (1)$$

where Q is defined by

$$Q(x) = 2(1-x)^2 \int_0^x \frac{t\Omega^2(t)}{(1-t^2)^2} dt + 2x \int_x^1 \frac{\Omega^2(t)}{(1+t)^2} dt - P(x-x^2), \quad (2)$$

and where P , and k (** Add physical properties of P and k **) are nonnegative physical parameters. Equations (1), (2) are supplemented by the boundary conditions

$$f(0) = 0, \quad \text{and} \quad \Omega(0) = 0, \quad \Omega(1) = 1. \quad (3)$$

The first goal of this paper is to present a numerical algorithm to obtain a solution to this boundary-value problem once P and k are specified. Our second goal is to investigate the dynamical system one obtains from the velocity field associated with f and Ω .

As suggested by Serrin in [1], our approach to obtaining an approximate solution to (1), (3) is to apply the Picard iteration scheme to this system; At each iteration the integral equation in (1)a, (3)a is first reduced to an initial-value problem and solved, then the boundary-value problem in (1), (3)b is addressed by applying a shooting method or a Galerkin scheme (both techniques will be employed in this paper). This entire process will be carried out in *Mathematica*. Once an initial guess of Ω is substituted in (2), the integrals in this term will be computed using the built-in function *NIntegrate*. Next, we solve the initial-value problem (1)a, (3)a using the built-in function *NDSolve*. The resulting f is then substituted in (1)b. Finally, we apply a shooting method to (1)b, (3)b, by combining *NDSolve* and *FindRoot*, to solve this boundary-value problem. This computation results in a new Ω . This algorithm is repeated until one reaches a fixed point. We will also apply this algorithm but replace the shooting method with the Galerkin scheme to illustrate the ease by which both methods can be implemented for this problem.

Once the above algorithm provides us with a solution pair (f, Ω) , we use this pair to obtain the velocity field of the flow, which in turn will lead to a dynamical system for particle motion. We use *NDSolve* to monitor the evolution of parcels of fluid to gain insight into the role of the parameters P and k and their influence in characterizing the nature of the vortex.

Section 2 contains a description of the derivation of (1), (3) from the Navier-Stokes equations. In Section 3 we present the details of the numerical algorithm including the code. In Section 4 we address the problem of parcel evolution under various parameter values. Section 5 contains a summary of our results.

2 Model Derivation

Here we present the various steps needed to arrive at (1), (3) from the steady-state Navier-Stokes equations for an incompressible viscous fluid, namely,

$$\mathbf{v} \cdot \nabla \mathbf{v} = -\frac{1}{\rho} \nabla p + \nu \Delta \mathbf{v}, \quad \text{div } \mathbf{v} = 0, \quad (4)$$

where \mathbf{v} , ρ and ν are the velocity, density and the kinematic viscosity of the fluid, respectively. The symbols ∇ , Δ and div represent the gradient, Laplace, and divergence operators, respectively.

Let (R, ϕ, θ) denote the spherical coordinates of a point of a point P , where $R > 0$ is the radial distance of P to the origin of the coordinate system, ϕ , taking values in $(0, \pi]$, is the angle between the position vector and the positive z -axis, and θ , taking values in $(0, 2\pi]$ is the meridian angle about the z -axis. The half-space domain is parameterized by the set of all (R, ϕ, θ) for which $R > 0$ and $0 < \phi \leq \frac{\pi}{2}$. The z -axis is described by $\phi = 0$ and the xy -plane by $z = \frac{\pi}{2}$.

Let $\{\mathbf{e}_R, \mathbf{e}_\theta, \mathbf{e}_\phi\}$ denote the standard orthonormal spherical basis at a point P , namely,

$$\begin{aligned} \mathbf{e}_R &= \langle \cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi \rangle, & \mathbf{e}_\theta &= \langle -\sin \theta, \cos \theta, 0 \rangle, \\ \mathbf{e}_\phi &= \langle \cos \theta \cos \phi, \sin \theta \cos \phi, -\sin \phi \rangle. \end{aligned} \quad (5)$$

In this basis we denote the components of the velocity field \mathbf{v} and its acceleration $\mathbf{a} = \mathbf{v} \cdot \nabla \mathbf{v}$ by (v_R, v_θ, v_ϕ) and (a_R, a_θ, a_ϕ) , respectively. The relation between the components of \mathbf{v} and \mathbf{a} can be found in many standard texts in fluid mechanics (including [5], p. 615). Despite the fact that these relations are well-known, we dedicate Appendix A to its derivation because of the programming challenge the transformation of the Navier-Stokes equations from rectangular to spherical coordinates offers. This effort parallels the one required for similar equations in geophysical fluid dynamics, whose derivation is not as well-known as in the present case, and yet the formalism we present in Appendix A generalizes in a straightforward fashion.

The equations in (4)a take the following form once the spherical basis in (5) is applied:

$$\begin{aligned} v_R \frac{\partial v_R}{\partial R} + \frac{v_\phi}{R} \frac{\partial v_R}{\partial \phi} &+ \frac{v_\theta}{R \sin \phi} \frac{\partial v_R}{\partial \theta} - \frac{v_\phi^2 + v_\theta^2}{R} = -\frac{1}{\rho} \frac{\partial p}{\partial R} + \\ &\nu \left(\nabla^2 v_R - \frac{2v_R}{R^2} - \frac{2}{R^2} \frac{\partial v_\phi}{\partial \phi} - \frac{2v_\phi \cot \phi}{R^2} - \frac{2}{R^2 \sin \phi} \frac{\partial v_\theta}{\partial \theta} \right), \\ v_R \frac{\partial v_\phi}{\partial R} + \frac{v_\phi}{R} \frac{\partial v_\phi}{\partial \phi} &+ \frac{v_\theta}{R \sin \phi} \frac{\partial v_\phi}{\partial \theta} + \frac{v_R v_\phi}{R} - \frac{v_\theta^2 \cot \phi}{R} = -\frac{1}{\rho R} \frac{\partial p}{\partial \phi} + \\ &\nu \left(\nabla^2 v_\phi + \frac{2}{R^2} \frac{\partial v_R}{\partial R} - \frac{v_\phi}{R^2 \sin^2 \phi} - \frac{2 \cot \phi}{R^2 \sin \phi} \frac{\partial v_\theta}{\partial \theta} \right), \\ v_R \frac{\partial v_\theta}{\partial R} + \frac{v_\phi}{R} \frac{\partial v_\theta}{\partial \phi} &+ \frac{v_\theta}{R \sin \phi} \frac{\partial v_\theta}{\partial \theta} + \frac{v_\theta v_R}{R} + \frac{v_\phi v_\theta \cot \phi}{R} = -\frac{1}{\rho R \sin \phi} \frac{\partial p}{\partial \theta} + \\ &\nu \left(\nabla^2 v_\theta - \frac{v_\theta}{R^2 \sin^2 \phi} + \frac{2}{R^2 \sin \phi} \frac{\partial v_R}{\partial R} + \frac{2 \cot \phi}{R^2 \sin \phi} \frac{\partial v_\phi}{\partial \theta} \right), \end{aligned} \quad (6)$$

where $\nabla^2 = \frac{\partial^2}{\partial R^2} + \frac{\partial^2}{\partial \phi^2} + \frac{\partial^2}{\partial \theta^2}$. Similarly, equation (4)b takes the form

$$\frac{1}{R^2} \frac{\partial(R^2 v_R)}{\partial R} + \frac{1}{R \sin \phi} \frac{\partial(v_\phi \sin \phi)}{\partial \phi} + \frac{1}{R \sin \phi} \frac{\partial v_\theta}{\partial \theta} = 0. \quad (7)$$

As suggested by Serrin in [1], we seek solutions of (6), (7) in the form

$$v_R = \frac{G(x)}{r}, \quad v_\phi = \frac{F(x)}{r}, \quad v_\theta = \frac{\Omega(x)}{r} \quad (8)$$

where F, G and Ω are as yet unknown, and the new variables x and r are defined as

$$x = \cos \phi, \quad r = R \sin \phi. \quad (9)$$

The symmetry imposed by the form of the solution in (8) is partially motivated by the boundary conditions along the z -axis, where we expect to have a line vortex, and the behavior of the solution as r approaches infinity, where we expect the velocity field to die out, and partially by our desire to develop a simple model of a tornado where we expect that the dependence of \mathbf{v} on θ is negligible, and thus absent in this point of view. The form in (8) also resembles closely the one suggested by Goldshtik in [2] where cylindrical coordinates constituted in the desired frame. Overall, this special ansatz is suggested by the study of universal solutions of Euler and Navier-Stokes equations, a description of which can be found in the recent monograph [3].

We first substitute (8) in the continuity equation (7) (see Appendix A) and arrive at the relation

$$G = F' \sin \phi. \quad (10)$$

Next we substitute (8) and (10) into (6). The resulting equations in F, Ω and p are

$$\begin{aligned} F'' + F'^2 + \frac{F^2 + \Omega^2}{\sin^2 \phi} &= \frac{R^3}{\rho} \frac{\partial p}{\partial R} + \nu(2F'' \cos \phi - F'' \sin^2 \phi), \\ FF' + \frac{(F^2 + \Omega^2) \cos \phi}{\sin^2 \phi} &= \frac{R^2 \sin \phi}{\rho} \frac{\partial p}{\partial \phi} + \nu F'' \sin^2 \phi, \\ F\Omega' &= \frac{R^2}{\rho} \frac{\partial p}{\partial \theta} - \nu \Omega'' \sin^2 \phi \end{aligned} \quad (11)$$

Since we seek solutions that are independent of θ the pressure term in (11)c vanishes and this equation reduces to

$$F\Omega' + \nu\Omega'' \sin^2 \phi = 0. \quad (12)$$

Next we eliminate p from (11)a-b by cross-differentiating the first equation with respect to ϕ and the second with respect to R and subtracting the resulting equations. Recalling that $x = \cos \phi$, this and (12) lead to the following system of equations in F and Ω :

$$\begin{aligned} \nu(1-x^2)F^{(iv)} - 4\nu x F''' + FF''' + 3F'F'' &= -\frac{2\Omega\Omega'}{1-x^2}, \\ F\Omega' + \nu(1-x^2)\Omega'' &= 0. \end{aligned} \quad (13)$$

These equations are supplemented with the five boundary conditions

$$\Omega(0) = F(0) = F'(0) = 0, \quad F(1) = 0 \quad \text{and} \quad \Omega(1) = 1. \quad (14)$$

Next we integrate the first equation in (13) three times with respect to x , reduce the triple integral to a single integral by performing integration by parts and using the boundary conditions in (14), and finally change dependent variables from F to f by

$$F = (1-x^2)f. \quad (15)$$

This results in (1).

3 Numerical Algorithm

We find an approximate solution to (1), (2), (3) by using a combination of the Picard iteration and the shooting method. Let P and k take positive values. The algorithm is as follows: a) We start with a guess for Ω , typically $\Omega(x) = x^\alpha$ for some α . With the function Q in (2) is now known up to a numerical integration, we solve the initial-value problem (1)a, (3)a using **NDSolve** in Mathematica as described in **Step 1**.

Step 1 (The Initial Value Problem to Construct f):

```
P=1; k=1;
xfinal = 0.9999;
omega[x_]=x;
solution1=NDSolve[{f'[x] == k^2/(1-x^2)^2*(2(1-x)^2*
  NIntegrate[t omega[t]^2/(1-t^2)^2, {t, 0, x},
  MaxRecursion -> 10] +
  2x NIntegrate[omega[t]^2/(1+t)^2, {t, x, xfinal},
  MaxRecursion -> 10]
  - P(x-x^2)) - f[x]^2, f[0] == 0}, f, {x, 0, xfinal}]
```

The boundary condition at $x = 1$ is replaced by $x = 0.9999$ to aid the internal function *NIntegrate* with the singularity at that point. The output of this program is an approximation, albeit a crude one, to the solution f of (1)a, (3)a. This expression is then substituted into (1)b, (3)b and the resulting boundary value problem is solved by a shooting method as follows:

Step 2 (The Boundary Value Problem - Shooting Method, to Establish $\Omega'(0)$):

```
Clear[omega]; newf[x_] = First[f[x] /. solution1];
output=FindRoot[First[Evaluate[omega[xfinal] /. NDSolve[{omega'[x]==omegap[x],
  omegap'[x]==-2 newf[x] omegap[x], omega[0]==0, omegap[0]==a},{omega, omegap},
  {x,0,xfinal}]]]-1, {a, 0.1, 0.9}];
```

Note the way **NDSolve** and **FindRoot** have been combined to implement the shooting method. Once this program accomplishes its task, we have in hand the appropriate initial condition $\Omega'(0)$, stored in **output**, which we now use to construct Ω accurately:

Step 3 (To Construct Ω):

```
Clear[omega];
solution2=NDSolve[{omega'[x]==omegap[x], omegap'[x]==-2 newf[x] omegap[x],
  omega[0]==0, omegap[0]==a /. output},{omega, omegap},
  {x,0,xfinal}, MaxSteps->10000, WorkingPrecision->15];
omega[x_]=First[omega[x] /. solution2];
```

The Ω just found will serve as an update for the original guess we used in **Step 1**. We are now ready to start the iteration process, namely, to execute steps one through three until we reach satisfactory convergence. A typical run takes about 50 iterations. The

convergence is surprisingly robust with respect to the initial guess and the parameter values k and P . Figure 5 shows the output of a typical run where the value of

$$\int_0^1 |F(x)|^2 + |G(x)|^2 + |\Omega(x)|^2 dx$$

is displayed (with 1 replaced by 0.9999) as a function of the number of iterations.

The shooting method is one way of computing an approximate solution to (1)b, (3)b. An alternative method is to use the Galerkin scheme because of the ease by which it can be programmed and because of its range of applicability in physical settings where one may confront instability with the shooting method. Referring to (1)b, (3)b, alternatively we seek a solution Ω in the form

$$\Omega = b(x) + \sum_{i=1}^N a_i \phi_i(x) \quad (16)$$

where ϕ_i s form a basis in $L^2(0,1)$ and satisfy zero boundary conditions at $x = 0$ and $x = 1$ while the function b is any function that satisfies the boundary conditions in (3)b. Typically we use $\phi_i = \sin i\pi x$. Next we substitute the expression in (16) into (1)b, take the inner product of the result with ϕ_i and end up with N simultaneous algebraic equations in the unknowns a_i s. This system is then solved using the standard linear solver in Mathematica. The main challenge in implementing this technique is making the output of step one, namely f , available to the integration routine in the inner product. We are fortunate in this problem in that because of the way we are implementing the iteration algorithm, the differential equation in (1)b is linear, albeit with non-constant coefficients. It turns out, however, that the Galerkin scheme as programmed below is considerably more general than in the setting we have introduced here and that with some care the Galerkin program generalizes to considerably more complex problems including nonlinear partial differential equations in multi-dimensional domains. We now list the alternate program whose output will be the approximate solution to the boundary value problem in (1)b, (3)b.

Alternate Steps 2 and 3 (To Construct Ω): The following program is written for $\phi_i(x) = \sin \pi x$, $b(x) = x$, $N = 10$, $P = 0.47$ and $k = 10$.

```

nn = 10; P = 0.47; k = 10; xfinal = 0.9999; phi[i_, x_] = Sin[i Pi
x]; phiprime1[n_, x_] = D[phi[n, x], x];
phiprime2[n_, x_] = D[phi[n, x], {x, 2}]; Omg[x_] = x + Sum[a[n] phi[n,
x], {n, 1, nn}]; newOmg[x_] = 1; (*first guess*) innerproduct[f_,
g_] := NIntegrate[f g, {x, 0, xfinal}, MaxRecursion -> 15];
Clear[a]; equations = Table[Sum[a[n]*innerproduct[phiprime2[n, x],
phi[i, x]] +
      2 a[n] innerproduct[f[x] phiprime1[n, x], phi[i, x]], {n, 1, nn}] ==
      -2 innerproduct[f[x], phi[i, x]], {i, nn}];
Clear[f]; vars = Table[a[n], {n, nn}]; sol = Solve[equations,
vars]; newOmg[x_] = First[Omg[x] /. sol];

```

Figures 1, 2, 3 and 4 show the output of the above programs for various values of P and k . These figures should be contrasted with the corresponding figures in [1] where Serrin

first offered a rigorous mathematical analysis of (1), (3) in terms of the two parameters P and k . In particular, Serrin discovered three dynamically different solutions to (1), (3) corresponding a) a vortex with an up-draft, where parcels of fluid approach the z -axis along xy -plane and leave domain along the positive z -axis, b) a vortex with a down-draft, where parcels of fluid are dragged down along the z -axis and leave the domain along the xy -plane and c) a vortex that displays an intermediate state, one in which parcels of fluid are drawn toward the z -axis along the xy -plane as well as downward along the z -axis, but leave the domain along a slanted line. Our algorithm captures all three states. In the next section we present a program that displays the evolution of parcels of fluids once each of the velocity fields in Figures 1, 2, 3 and 4 are determined.

4 Dynamical Systems and Flow Visualization

A typical output of the programs in the previous section is a velocity field in the form (v_R, v_ϕ, v_θ) . This representation, which is expressed in spherical coordinates, defines the dynamical system

$$\frac{dR}{dt} = v_R(R, \phi, \theta), \quad \frac{d\phi}{dt} = \frac{1}{r}v_\phi(R, \phi, \theta), \quad \frac{d\theta}{dt} = \frac{1}{r \sin \phi}v_\theta(R, \phi, \theta), \quad (17)$$

where $(R(t), \phi(t), \theta(t))$ represents the parametrization of a path of a particle under the action of the velocity field \mathbf{v} . Once the system of differential equations in (17) is solved numerically, we can plot the path of a typical particle in order to visualize the action of the velocity field. In particular, by plotting the trajectories of an ensemble of particles whose initial configuration is a familiar geometric figure, say a sphere, we can track the deformation of the parcel and its direction of motion (updraft or downdraft) visually.

The only challenge in completing this visualization task is the management of the data one has obtained from computing the velocity field numerically, as described in the previous section, while solving the system in (17) whose right-hand side is given numerically, and, the last step, to collect the information from individual particles in the ensemble at each time slot to display the snapshot of the flow. The following program accomplishes these steps and displays a set of snapshots that can readily be animated to provide a convincing picture of the flow.

Visualization Program

```
Needs["Graphics`Graphics`"];
Needs["Graphics`Legend`"];
e1 = .1; e2 = .1; e3 = 3; rad = .03;
n = 12; k = 3.5; P = 0.442;
xfinal = 0.9999;
time = 5; NoofSnapshots = 12; deltat = N[time/NoofSnapshots];
parcels = 3;
colors = {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]};
filament = Graphics3D[Line[{{0, 0, 0}, {0, 0, 3}}]];
(*Computing the Velocity Fields*)
Print["Computing the Velocity Field"];
```

```

omegaEqn[a_] :=
  NDSolve[{omega'[x] == omegap[x], omegap'[x] == -2 newf[x] omegap[x],
    omega[0] == 0, omegap[0] == a}, {omega, omegap}, {x, 0, xfinal},
    MaxSteps -> 10000, WorkingPrecision -> 15];
Do[ClearAll[F, derF, omega, newf, f, Va, Vr, Vt]; Print["jj = ", jj];
  Clear[f, omega];
  k = k + 0.5;
  omega[x_] = x;
  Do[solution1 =
    NDSolve[{f'[x] ==
      k^2/(1 - x^2)^2*(2(1 - x)^2*
        NIntegrate[t omega[t]^2/(1 - t^2)^2, {t, 0, x},
          MaxRecursion -> 10] +
        2x NIntegrate[omega[t]^2/(1 + t)^2, {t, x, xfinal},
          MaxRecursion -> 10] - P(x - x^2)) - f[x]^2,
      f[0] == 0}, f, {x, 0, xfinal}];
    newf[x_] = First[f[x] /. solution1];
    Clear[omega];
    output =
      FindRoot[
        First[Evaluate[omega[xfinal] /. omegaEqn[a]]] - 1, {a, 0.1, 0.9}];
    Clear[omega];
    solution2 = omegaEqn[a /. output];
    omega[x_] = First[omega[x] /. solution2], {j, 1, 20}];
F[x_] = 1/k(1 - x^2)newf[x]; derF[x_] = D[F[x], x];
Va[alpha_, theta_, r_] = (F[Cos[alpha]])/(r Sin[alpha]);
Vr[alpha_, theta_, r_] = (derF[Cos[alpha]])/(r);
Vt[alpha_, theta_, r_] = (omega[Cos[alpha]])/(r Sin[alpha]);
Print["Velocity Field is Computed"];
diffeqn[{b_, c_, d_}] :=
  NDSolve[{alpha'[t] == (1/r[t]) Va[alpha[t], theta[t], r[t]],
    theta'[t] == 1/(r[t] Sin[alpha[t]]) Vt[alpha[t], theta[t], r[t]],
    r'[t] == Vr[alpha[t], theta[t], r[t]], alpha[0] == b, theta[0] == c,
    r[0] == d}, {alpha, theta, r}, {t, 0, time}, MaxSteps -> 100000,
    WorkingPrecision -> 15];
CoordPointsCart =
  Flatten[Table[{x -> N[e1 + 0.1 + rad*Sin[v] Sin[w]],
    y -> N[e2 + rad*Sin[v] Cos[w]], z -> N[e3 + rad*Cos[v]]}, {w, 0,
    2 Pi, Pi/n}, {v, 0, Pi, Pi/n}], 1];
PointsCart = {x, y, z} /. CoordPointsCart;
SpherePoints = {N[ArcCos[z/Sqrt[x^2 + y^2 + z^2]]],
  N[ArcCos[y/Sqrt[x^2 + y^2]]], N[Sqrt[x^2 + y^2 + z^2]]} /.
  CoordPointsCart;
MakePointsCart = Table[Point[PointsCart[[i]]], {i, Length[PointsCart]};
graph = Graphics3D[{colors[[jj]], MakePointsCart}];

```

```

Print["Solving IVP ...."];
oldsolution = Flatten[diffeqn /@ SpherePoints, 1];
Print["Finished with NDSolve ..."];
X[t_] := {r[t] Sin[alpha[t]] Sin[theta[t]],
          r[t] Sin[alpha[t]] Cos[theta[t]], r[t] Cos[alpha[t]]];
FF[t_] = X[t] /. oldsolution;
Do[tt[i] = (i - 1)*deltat;
   snapshot[jj, i] =
     Show[{filament}, Graphics3D[{colors[[jj]], Point /@ FF[tt[i]]}],
        PlotRange -> {{-1, 1}, {-1, 1}, {0, 4}},
        PlotLabel ->
          StringJoin["k = ", ToString[k], ", P = ", ToString[P], ", t = ",
                    ToString[tt[i]]], {i, 1, NoofSnapshots}, {jj, 1, 3}];
Table[snap[i] = Table[snapshot[jj, i], {jj, 1, 3}], {i, 1, NoofSnapshots}];
Table[Show[snap[i]], {i, 1, NoofSnapshots}];

```

This program is written to monitor the evolution of the same parcel of fluid, a sphere of radius 0.03 centered at (0.1, 0.1, 3), under the action of \mathbf{v} when $P = 0.442$ while k takes on three values at 4, 4.5 and 5. We identify 325 points on the surface of the sphere whose evolution are monitored over the time interval (0, 10). Basic mathematical formulas are used to enable one to represent the position of each particle in rectangular and spherical coordinates as needed.

Figures 6 – 9 show the output of this program at various instances of time. The values of P and k chosen are in the down-draft regime so, as expected, each parcel is sucked toward the vortex axis and pushed downward. Particles are colored according to their value of k , red corresponding to $k = 4$ and blue to $k = 5$. and . These figures show that the lower the viscosity ν (which is related to k according to $k = \frac{1}{2\nu}$), the faster the parcel is dragged down the z -axis. Eventually, the parcel with the higher viscosity passes through the one with lower viscosity as the parcel approaches the xy -plane. All parcels then leave the neighborhood around the z -axis acquiring the shape of a cone.

Figures 10 – 13 are the output of a slightly different program in which P and k are kept fixed while the three parcels are positioned relatively close to each other and colored according to their initial locations. Thus these figures demonstrate the degree by which the space is deformed about the z axis as time evolves.

5 Conclusion

In the previous sections we presented an algorithm for obtaining the solution to a system of nonlinear integro-differential equations that model fluid flows exhibiting tornadic behavior. These equations were derived from the full Navier-Stokes equations under certain symmetry assumptions. The algorithm consisted of applying the Picard iteration (fixed point) method together with a boundary-value solver (shooting as well as the Galerkin method). The main feature of our effort is that we carry out all of our computations in *Mathematica*. The velocity field obtained by this technique is then the subject

of our investigation as parameter values are varied and the structure of the resulting fluid flows are visualized.

Our point of departure required transforming the standard Navier-Stokes equations to spherical coordinates. A program was presented in Appendix A to accomplish this task. The significance of this program is that it can be readily generalized to other systems of partial differential equations and coordinate systems. Next we extended the functionality of the program and applied it to the special ansatz describing the class of velocity fields that include tornadic motions. After eliminating the pressure term from the latter system, we arrived at the afore-mentioned system of integro-differential equations.

We experimented with several parameter values to demonstrate that the convergence of our iterative algorithm was rapid. The velocity field we obtained in this manner is written in terms of standard interpolation functions available in *Mathematica*. Each velocity field defines a three-dimensional dynamical system whose trajectories are the orbits in the fluid flow. Three types of motions are observed, those in which fluid parcels are sucked into the funnel and dragged upward (Figure 1), those in which fluid parcels are pulled down by the line vortex and driven away from the vortex along the xy -plane (Figure 2), and an intermediate flow in which fluid parcels are sucked in along the xy -plane and the line vortex but leave the domain along a slanted line (Figure 3). These three motions were predicted in the original paper of Serrin [1].

A future line of investigation is to understand the stability of each of the three flows as a solution of the full Navier-Stokes equations. Such a study requires linearizing the Navier-Stokes equation about each of these steady-state solutions. Although the resulting equations will be linear, they possess nonconstant coefficients. The challenge then is to characterize the spectrum of the linearized system. In a future effort, we will present this analysis within the context of *Mathematica*.

6 Appendix A

In this section we present the program that delivers the Navier-Stokes equations in spherical coordinates. Although these equations are available in several texts, we present this program for the sake of completion as well as providing the blueprint for similar derivation that are not so readily available. In that category, we mention the equations of motion of the geophysical fluid dynamics in various geometrical symmetries. The following program begins with the Navier-Stokes equations in rectangular coordinates, transforms them into spherical coordinates and then applies the special ansatz in (8) to derive the equations in (11) and (12).

Our strategy is very simple. We have two equivalent representation of the velocity field, in rectangular and spherical coordinates. So we begin with the Navier-Stokes equations in rectangular coordinates, substitute the spherical representation of \mathbf{v} in component form into these equations, simplify as necessary, and finally solve for $\{\frac{\partial v_R}{\partial t}, \frac{\partial v_\phi}{\partial t}, \frac{\partial v_\theta}{\partial t}\}$. The key to the success of this idea is in the various applications of the simplification routines (such as **Simplify**, **ExpandAll**, **PowerExpand**) and the implementation of appropriate substitutions at the right time.

Coordinate Transformation Program:

```

rect := {x -> R Cos[th] Sin[phi], y -> R Sin[th]Sin[phi],
        z -> R Cos[phi]};
spherical := {R -> Sqrt[x^2 + y^2 + z^2], th -> ArcTan[y/x],
             phi -> ArcCos[z/Sqrt[x^2 + y^2 + z^2]]};
domain = {Sqrt[x^2 + y^2 + z^2] -> R, ArcTan[y/x] -> th,
         ArcCos[z/Sqrt[x^2 + y^2 + z^2]] -> phi};
eR = {Cos[th]Sin[phi], Sin[th]Sin[phi], Cos[phi]}; eth =
{-Sin[th], Cos[th], 0}; ephi = {Cos[th]Cos[phi], Sin[th]Cos[phi],
-Sin[phi]}; pres = p[t, R, th, phi]; vel = w[t, R, th, phi] eR +
u[t, R, th, phi] eth +
v[t, R, th, phi] ephi;
pressure = pres /. spherical; vrect = vel /. spherical; var = {x,
y, z}; divergence = ExpandAll[Sum[D[vrect[[i]], var[[i]]], {i,
3}]] // domain; divergence = divergence /. rect; divergence =
Simplify[PowerExpand /@ Simplify /@ divergence]; // Timing acc =
Table[
    D[vrect[[i]], t] + Sum[vrect[[j]] D[vrect[[i]], var[[j]]], {j, 3}] +
    1/R D[pressure, var[[i]]] -
    nu Sum[D[vrect[[i]], {var[[j]], 2}], {j, 3}], {i, 3}];
acc = ExpandAll[acc] // domain; newacc = acc /. rect; // Timing
newsimplify[x_] := PowerExpand[Simplify[x]]; newacc =
Table[newsimplify /@ newacc[[i]], {i, 3}]; // Timing eqns =
Table[newacc[[i]] == 0, {i, 3}]; newvelt = {D[w[t, R, th, phi],
t], D[u[t, R, th, phi], t],
    D[v[t, R, th, phi], t]};
NavierStokesSpherical =
    Simplify /@ ExpandAll[Solve[eqns, newvelt]]; // Timing
ansatz = {w[t_, R_, th_, phi_] = G[Cos[phi]]/(R Sin[phi]),
v[t_, R_, th_, phi_] = F[Cos[phi]]/(R Sin[phi]),
u[t_, R_, th_, phi_] = Omega[Cos[phi]]/(R Sin[phi]),
p[t_, R_, th_, phi_] = P[R, phi]};
ConsOfMass = Solve[divergence == 0, G[Cos[phi]]]; neweqns =
Simplify[Table[NavierStokesSpherical[[1, i, 2]], {i, 3}]];
replacement = {Cos[phi] -> x, Sin[phi] -> Sqrt[1 - x^2],
    Cot[phi] -> x/Sqrt[1 - x^2], Csc[phi] -> 1/Sqrt[1 - x^2]};
G[x_] = First[G[Cos[phi]] /. ConsOfMass] /. replacement; eqns =
Numerator[PowerExpand[Together[Simplify[eqns]]]]; DPs = First[
    Solve[{eqns[[2]] == 0, eqns[[3]] == 0}, {D[P[R, phi], R],
    D[P[R, phi], phi]}];
neweqn = D[DPs[[1, 2]], phi] - D[DPs[[2, 2]], R]; neweqn =
Simplify[Numerator[Together[TrigExpand[neweqn] /. replacement]]];

```

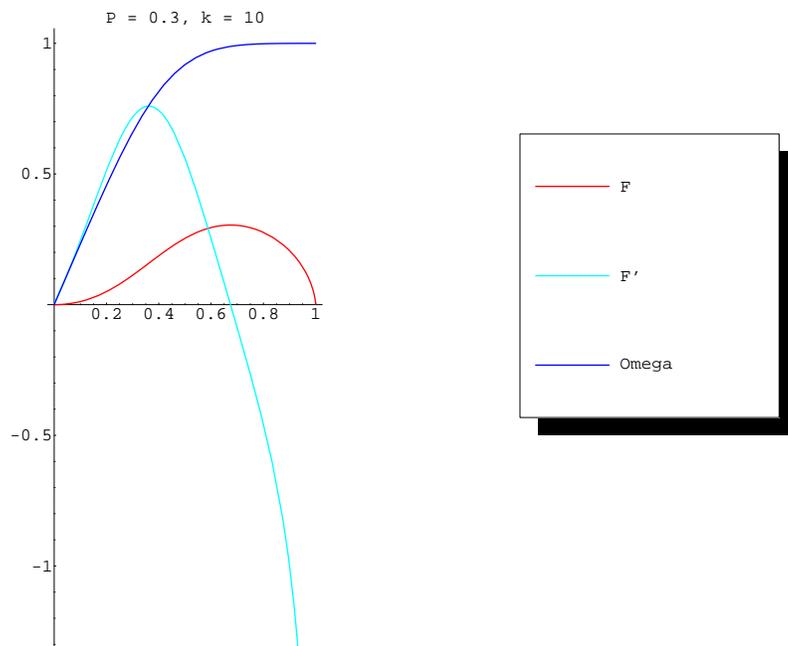


Figure 1: The functions F , F' and Ω when $k = 10$ and $P = 0.3$.

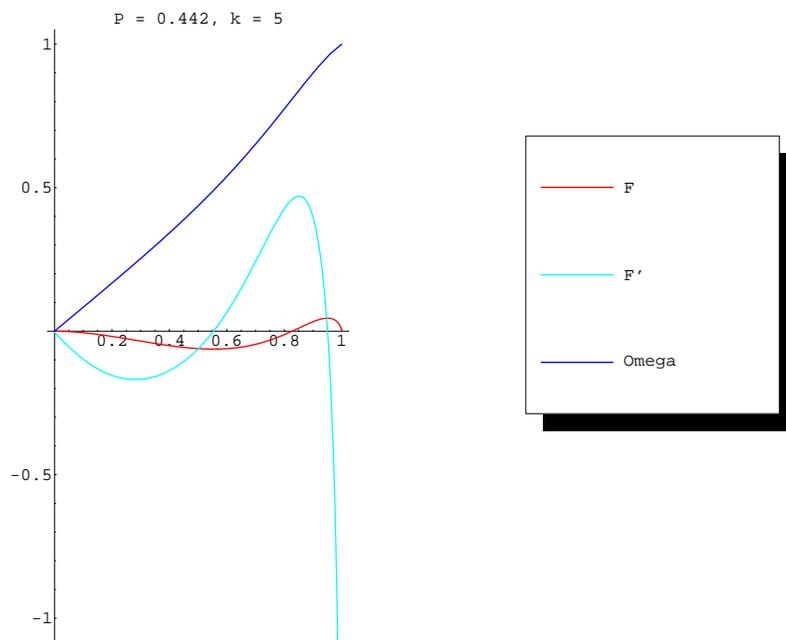


Figure 2: The functions F , F' and Ω when $k = 5$ and $P = 0.442$.

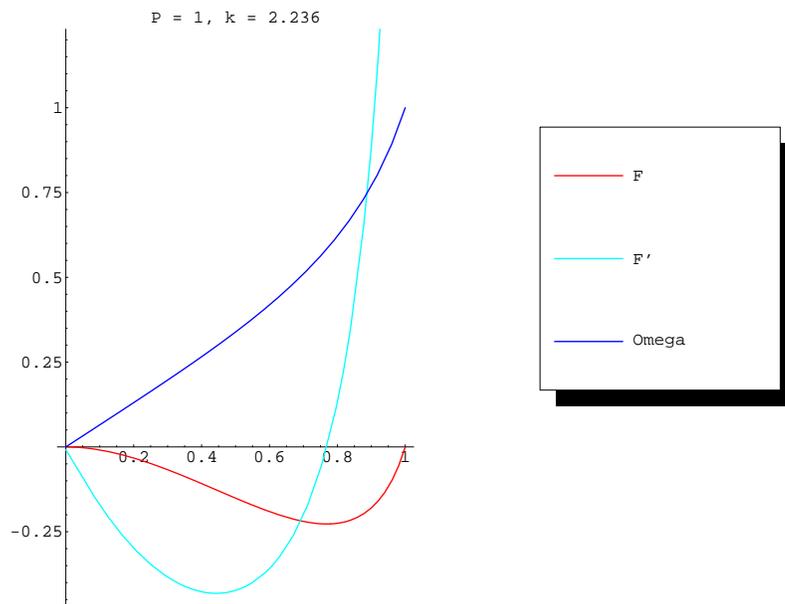


Figure 3: The functions F , F' and Ω when $k = 2.236$ and $P = 1$.

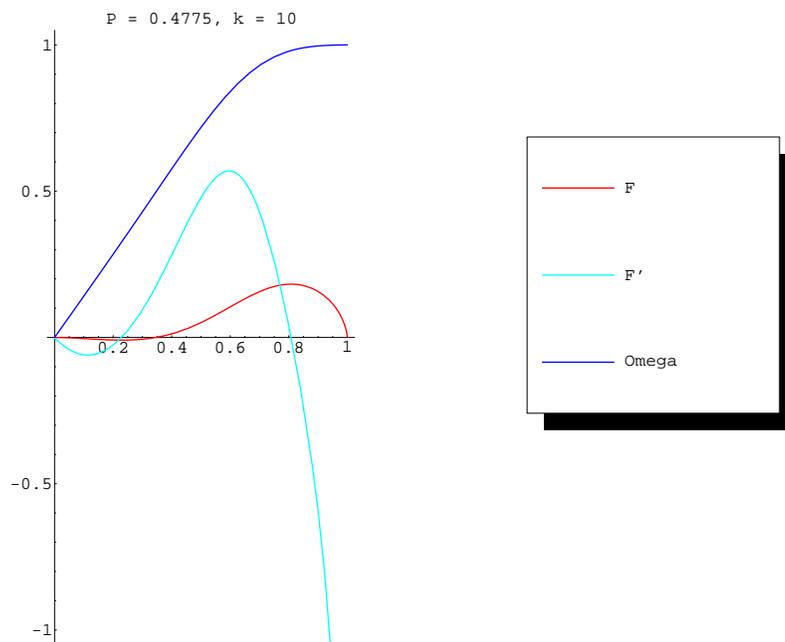


Figure 4: The functions F , F' and Ω when $k = 10$ and $P = 0.4775$.

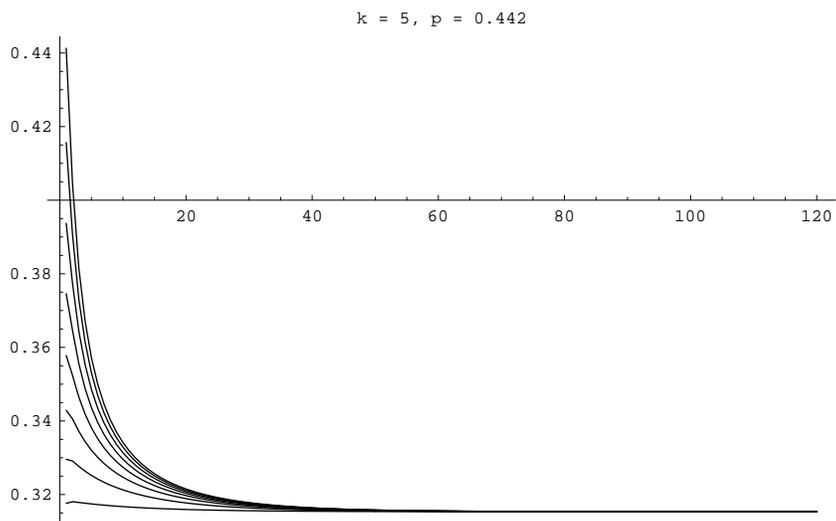


Figure 5: The graph of $\int_0^{0.9999} |F(x)|^2 + |G(x)|^2 + |\Omega(x)|^2 dx$ as a function of iterations in the algorithm.

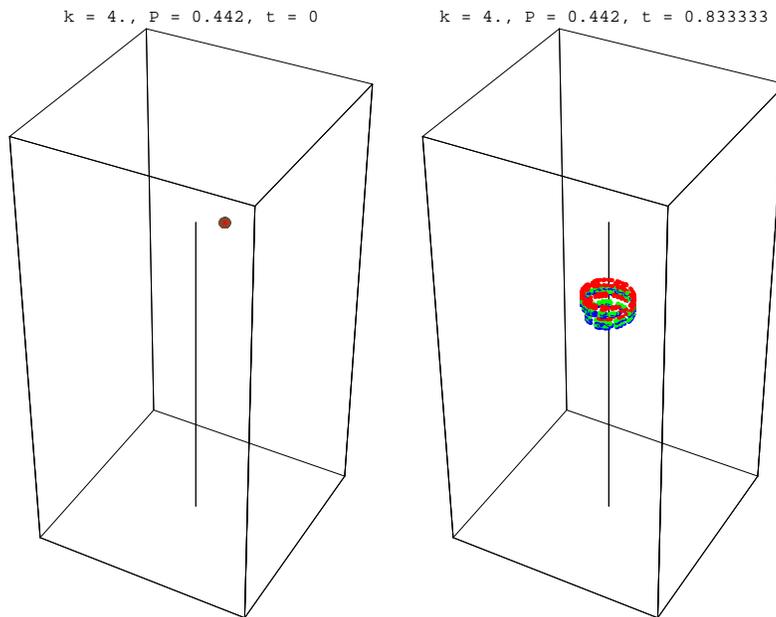


Figure 6: Flow of a parcel of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 4, 4.5$ and 5 .

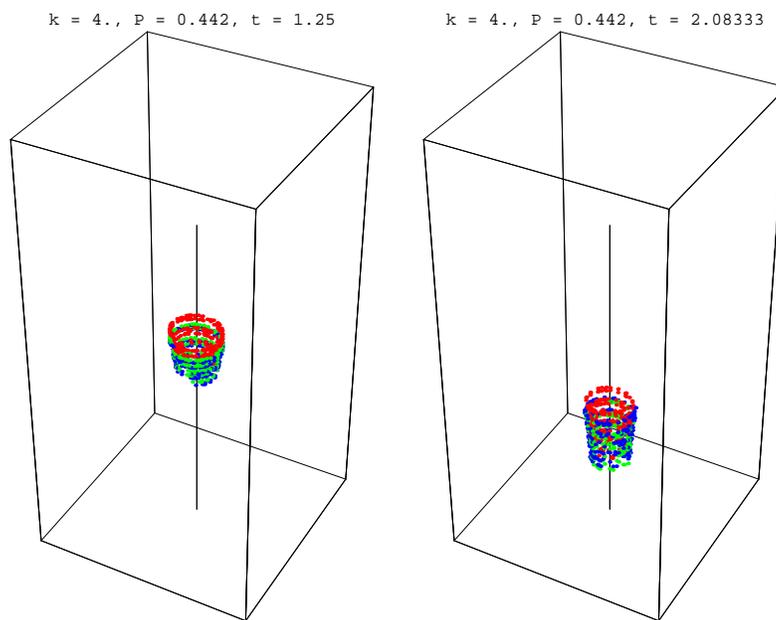


Figure 7: Continued: Flow of a parcel of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 4, 4.5$ and 5 .

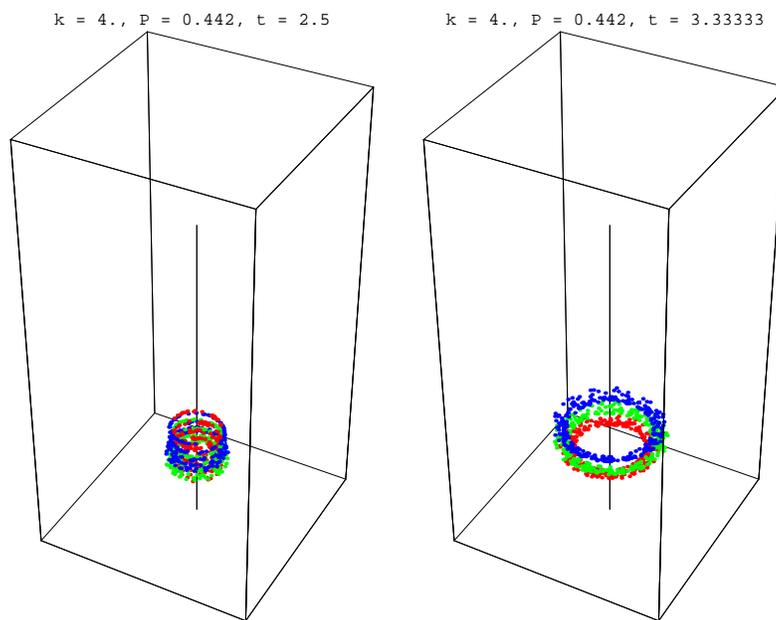


Figure 8: Continued: Flow of a parcel of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 4, 4.5$ and 5 .

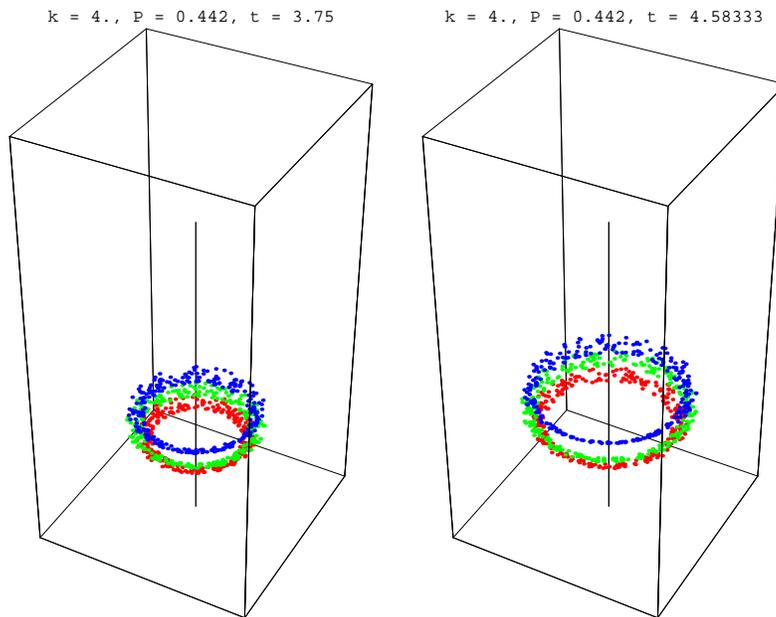


Figure 9: Continued: Flow of a parcel of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 4, 4.5$ and 5 .

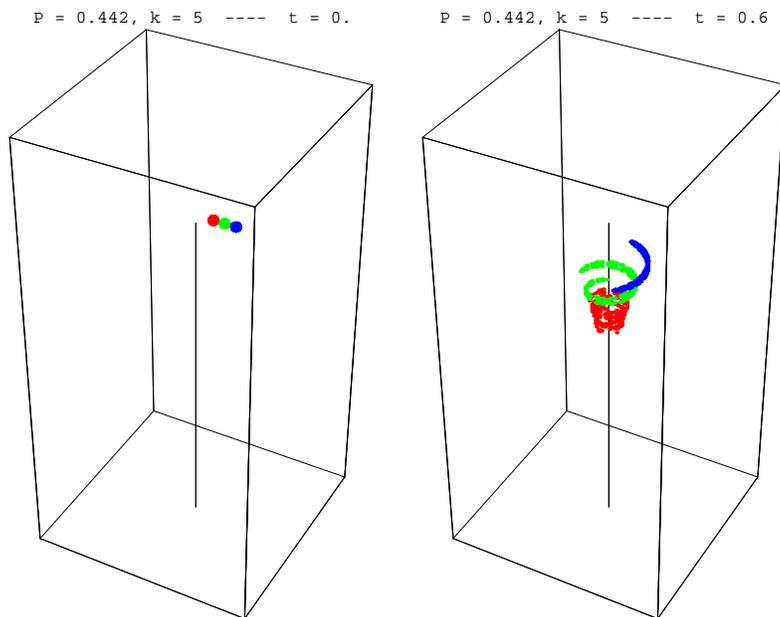


Figure 10: Flow of three parcels of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 5$.

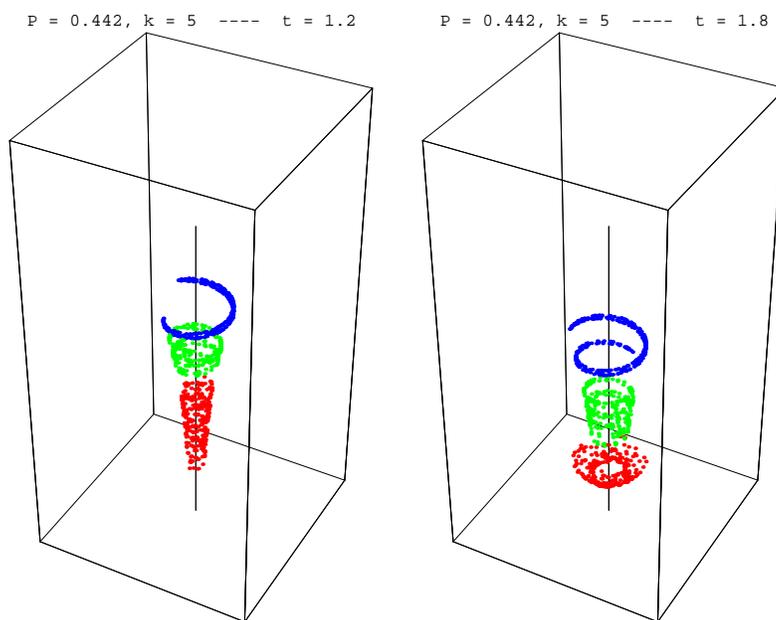


Figure 11: Continued: Flow of three parcels of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 5$.

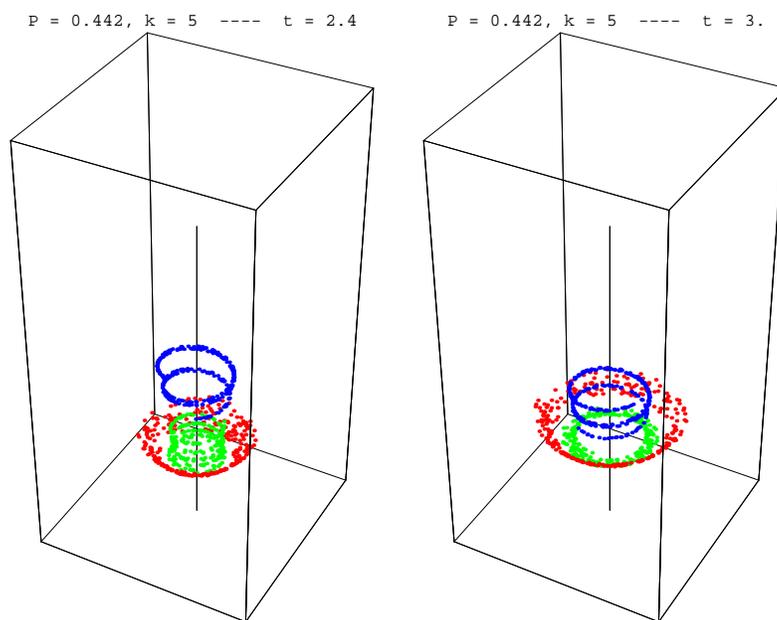


Figure 12: Continued: Flow of three parcels of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 5$.

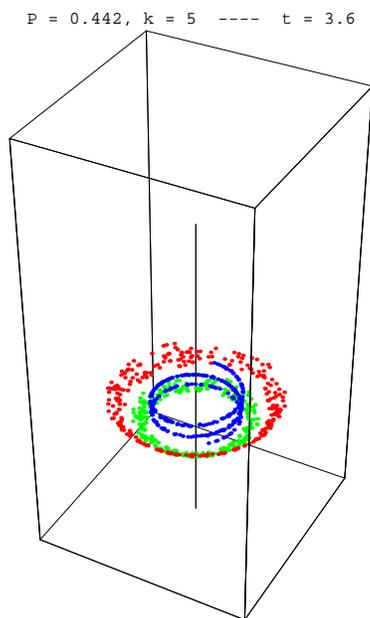


Figure 13: Continued: Flow of three parcels of fluid under the dynamics defined by (F, F', Ω) with $P = 0.442$ and $k = 5$.