# LAGRANGIAN COHERENT STRUCTURES OF ACCUMULATING INERTIAL PARTICLES IN VORTEX FLOW ON A DISK

---

Internship at

San Diego State University

---

by

Erwin Zeekant

fall 2011

# SAN DIEGO STATE UNIVERSITY

Erwin Zeekant

s0141046

San Diego State University

Department of Aerospace Engineering and Engineering Mechanics

prof.dr.ir. G.B. Jacobs, supervisor

San Diego, CA

United States of America

September - December 2011

University of Twente

Faculty of Engineering Technology (CTW)

Engineering Fluid Dynamics

prof.dr.ir. H.W.M. Hoeijmakers, UT-supervisor

# ABSTRACT

Lagrangian coherent structures of accumulating inertial particles
in vortex flow on a disk
by
Erwin Zeekant
San Diego State University, 2011

In this study a numerical investigation into the accumulation of inertial particles in vortex flow on a disk and its Lagrangian Coherent Structures is presented. A numerical simulation of inertial and gas particles, as in a gas-particle separator with a region of high vorticity, is performed. The structures can be shown by calculating the Finite-time Lyapunov Exponent (FTLE) for each grid point. This is a measure for the amount of stretching of particles at a certain location. By integrating forward in time, the forward FTLE can be calculated and by integrating backward in the the backward FTLE. Where the forward FTLE field show repelling Lagrangian Coherent Structures and the back FTLE field the attracting Lagrangian Coherent Structures.

A MATLAB code is written to solve the equations of motion of the inertial and gas particles and which immediately calculates the corresponding FTLE fields at that time step. The code is tested for a simple case and is used to analyse the vortex flow. For low Stokes numbers the accumulation is very slow, but when this is increased the accumulation becomes much quicker. Above a critical value of the Stokes number, where the inertial particles will not accumulate anymore and all will move to the wall. There is no attractor or repeller found which should lead to the accumulation of the particles in the FTLE fields. The particles follow the structures that can be seen, but these are not leading to the accumulation.

The results are compared to the results which are obtained by the drift flux model. With this model it is possible to give an analytical flow field for the inertial particles. For low Stokes numbers the results show good similarity. When the Stokes number is increased, the paths of the particles do not match anymore, but still give a good approximation for the accumulation point. When the Stokes number is too high the accumulation point and the paths do not show good similarity.

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| Symbol | Description | Unit |
|---|---|---|
| $d$ | Diameter | $m$ |
| $r$ | Radial position | $-$ |
| $t$ | Time | $-$ |
| $u$ | Velocity | $-$ |
| $\mathbf{x}$ | Position vector in Cartesian coordinates | $-$ |
| $F$ | Force | $N$ |
| $R$ | Characteristic length | $m$ |
| $\mathbf{S}$ | Rate of strain tensor | $-$ |
| $St$ | Stokes number | $-$ |
| $T$ | Integration length | $-$ |
| $V$ | Volume | $m^3$ |
| $\eta$ | x-position in co-rotating frame | $-$ |
| $\theta$ | Angular position | $rad$ |
| $\dot{\theta}$ | Angular velocity | $\frac{rad}{s}$ |
| $\mu$ | Dynamic viscosity | $\frac{kg}{ms}$ |
| $\xi$ | y-position in co-rotating frame | $-$ |
| $\rho$ | Density | $\frac{kg}{m^3}$ |
| $\sigma_{t_0}^{T}$ | Finite-time Lyapunov exponent | $-$ |
| $\tau$ | Relaxation time | $s$ |
| $\phi$ | Angle for co-rotating frame | $rad$ |
| $\phi_{t_0}^{t_0+T}$ | Positions after being affected by a flow | $-$ |
| $\Delta$ | Cauchy-Green deformation tensor | $-$ |
| $\Gamma$ | Characteristic circulation | $\frac{m^2}{s}$ |
| $\Psi$ | Stream function | $-$ |

| Subscripts | Description |
|---|---|
| $d$ | Drag |
| $g$ | Gas particles |
| $i$ | Mirror image of vortex in cylinder |
| $m$ | Mixture |
| $p$ | Inertial particle |
| $r, \theta$ | Polar coordinates |
| $v$ | Vortex in cylinder |
| $x, y$ | Cartesian coordinates |
| $0$ | Initial |

| Other | Description |
|---|---|
| $I$ | Identity matrix |
| $\lambda$ | Eigenvalue |
| $\delta$ | Difference |
| $\nabla$ | Gradient |
| $\|M\|_2$ | $L_2$ norm |

# LIST OF TABLES

PAGE

# LIST OF FIGURES

PAGE

# ACKNOWLEDGEMENTS

I would like to thank prof. Jacobs for giving me the opportunity to do my internship at San Diego State University and for his support and advice during my internship. I'm also thanking prof. Hoeijmakers for his help finding an internship and dr. Hagmeijer for his support during the project. Furthermore I would like to thank my roommates Daniel and Sonia for letting me stay with them and making it a great experience. Finally, I would like to thank my girlfriend Anne for encouraging me to do my internship abroad.

# CHAPTER 1
# INTRODUCTION

To separate small particles or small liquid droplets from a gas, gas-particle and gas-condensate separators can be used. Instead of filters it makes use of vortex separation. The separators consist of a cylindrical tube which contains a region of high vorticity, which can be seen as a steady helical vortex filament in the cylinder.

The application of a three dimensional helix filament has already been investigated by Hardin [3]. If the pitch of the helix is very large compared to the radius of the tube, the three dimensional effects can be neglected. The flow field for the two dimensional case is described by Lamb [5]. A point vortex is placed in the circular domain and an image vortex is placed outside the domain to satisfy the boundary conditions at the wall. Due to the image vortex, the flow field becomes a steady circular motion. If a co-rotating frame is used, the flow becomes steady and is described by a time-independent stream function. Further analysis for the two dimensional case is done by IJzermans and Hagmeijer [4]. A potential flow is used to describe the flow field for one or more vortices in the domain. The results show the accumulation of the inertial particles. The accumulation point of the inertial particles are solved analytically and show good agreement with the calculations.

In the present study the motion of inertial particles is further analysed using finite-time Lyapunov exponents (FTLE). With FTLE's it is possible to show Lagrangian Coherent Structures (LCS) for a flow. Haller [1] determined transport barriers in the flow field using a Lagrangian frame. These barriers are lines in two-dimensional flows and surfaces in three-dimensional flow which the fluid particles can not cross. If the FTLE is computed by integrating the trajectories of the fluid particles over a time. If the integration is done backward in time, this will reveal attracting barriers. When the trajectories are integrated forward in time, this will reveal the repelling barriers. The backward and forward FTLE's identify the LCS. Shadden *et al.* [9] implemented the method of Haller for two dimensional aperiodic flows. In their study it is shown how the FTLE can be computed. Three examples are shown for which they computed the LCS. The first example is a simple double-gyre flow, for which it is obvious to see there is a separatrix which should be leading to a repelling barrier. The computed FTLE field shows agreement with this expectation.

Sapsis and Haller [8] describe a criteria for the global attraction of finite size inertial particles. Sapsis and Haller [2] showed earlier that particles with very low Stokes numbers converge to a slow manifold. Passive tracers are always affected by the FTLE field, but

inertial particles can behave differently. With the criteria it is possible to make a distinction between areas that are affected by by the FTLE field and areas which are not affected by the FTLE field. The criteria is depending on the Stokes number of the inertial particles and the strain-rate tensor in the flow field. For a given steady flow field of the fluid it is only depending on the Stokes number, because the strain-rate tensor becomes a constant.

For low Stokes numbers the motion and FTLE's can also be calculated by using the drift flux model. The relative velocity between the gas particles and inertial particles can be derived by combining the momentum equations for the mixture and the dispersed phase. This was done by Manninen *et al.* [6].

The goal of this study is to compute the Lagrangian coherent structures for a vortex flow on a disk. With the FTLE fields it could possible to look more in detail to the accumulation of the inertial particles. To do this the passive tracers and inertial particles are traced and with these results the FTLE field can be computed. A MATLAB code is written to solve the equations of motions and immediately compute the corresponding FTLE field at that time step.

The physical model with its flow field and the used methods are explained in Chapter 2. Different methods are used to trace the particles and the computing of the FTLE field is described. Furthermore a criteria for global attraction of inertial particles is discussed. In Chapter 3 some necessary parameters will be determined and the MATLAB code will be compared to an analytical solution for a simple case. In Chapter 4 the computed FTLE's will be shown and the results will be compared to the results from the drift flux model. Chapter 5 contains the final conclusions and the recommendations.

# CHAPTER 2
# METHODOLOGY

The goal is to obtain the FTLE field for a potential flow. To do so, a MATLAB code is developed to solve the equations of motion for this potential flow and at each time step in the calculation the FTLE field can be determined. These calculations can be done for both gas and inertial particles. The equations of motion of the gas particles are solved using the velocity field, which is described analytically. The particle tracking of the gas is done using a fourth order Runge-Kutta method and for the inertial particles, the three steps Adams-Bashforth method is used. The flow can be calculated using a rotating frame or a co-rotating frame, where for a co-rotating frame the stream function becomes time-independent.

In this chapter, the physical model and the corresponding flow are described in section 2.1. In section 2.2 the finite-time Lyapunov exponent is explained and it is shown how it can be calculated. To do this, it is necessary to calculate the final positions of the particles, which is done by the methods in section 2.3. The FTLE field is not at every point affecting the flow of the inertial particle. With the criteria described in 2.4, it is possible to distinct areas where the particles respond to it and where they do not.

## 2.1 FLOW FIELD

The flow field is generated by a point vortex at a point in the circular domain with radius $R$. The vortex is not only rotating at one point, but is also moving around the centre of the domain. The point vortex has a radial position of $r_v$ and an angle of $\theta_v(t)$. The strength of the vortex is given by $\Gamma_v$. The physical model is shown in Fig. 2.1. By choosing $R$ as the characteristic length and $\Gamma_v$ as the characteristic circulation, all other variables are made dimensionless. The dimensionless time can be introduced by the characteristic time scale $R^2/\Gamma$. The dimensionless radial position of the point vortex is $r_v = 0.5$ in the analysis. To satisfy the boundary conditions, zero normal velocity at $r = 1$, it is necessary to place a counter rotating vortex, with strength $\Gamma_i = -\Gamma_v$, outside the domain at position $(r_v^{-1}, \theta_v(t))$ which was described by Milne-Thomson [7]. Therefore the dimensionless stream function for this problem consist of a term for the vortex in the cylinder and its mirrored vortex outside the cylinder:

$$\Psi(r,\theta) = \Gamma_v(\Psi_v(r,\theta,r_v,\theta_v(t)) - \Psi_i(r,\theta,r_v,\theta_v(t))) \tag{2.1}$$

With the stream function for the point vortex in the domain as:

$$\Psi_v(r,\theta,r_v,\theta_v(t)) = -\frac{1}{4\pi}\ln(r^2 + r_v^2 - 2rr_v\cos(\theta - \theta_v(t))) \tag{2.2}$$

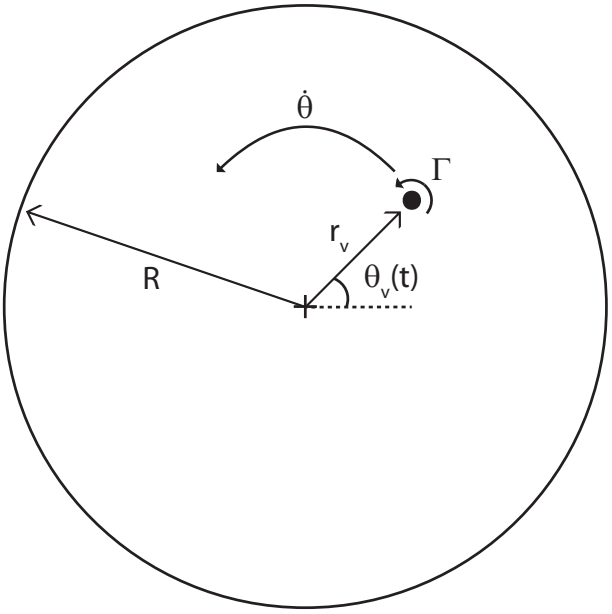



**Figure 2.1. Circular domain with a point vortex at $(r_v, \theta_v(t))$ rotating around the centre of the domain.**

and:

$$\Psi_i(r, \theta, r_v, \theta_v(t)) = \Psi_v(r, \theta, r_v^{-1}, \theta_v(t)) \tag{2.3}$$

The position and the angular velocity of vortex rotating around the centre of the domain are respectivaly:

$$\theta_v(t) = \frac{1}{2\pi}\frac{1}{1 - {r_v}^2}t \tag{2.4}$$

$$\dot{\theta}_v = \frac{1}{2\pi}\frac{1}{1 - {r_v}^2} \tag{2.5}$$

The velocity field can be obtained from the stream function using:

$$u_r = \frac{1}{r}\frac{\partial \Psi}{\partial \theta}, \quad u_\theta = -\frac{\partial \Psi}{\partial r} \tag{2.6}$$

The velocities then become:

$$\begin{aligned} u_r(r, \theta, r_v, \theta_v(t)) &= -\frac{r_v \sin(\theta - \theta_v(t))}{2\pi(r^2 + {r_v}^2 - 2rr_v\cos(\theta - \theta_v(t))} + \frac{r_v \sin(\theta - \theta_v(t))}{2\pi(1 + r^2{r_v}^2 - 2rr_v\cos(\theta - \theta_v(t))} \\ u_\theta(r, \theta, r_v, \theta_v(t)) &= \frac{r - r_v\cos(\theta - \theta_v(t))}{2\pi(r^2 + {r_v}^2 - 2rr_v\cos(\theta - \theta_v(t))} - \frac{r{r_v}^2 - r_v\cos(\theta - \theta_v(t))}{2\pi(1 + r^2{r_v}^2 - 2rr_v\cos(\theta - \theta_v(t))} \end{aligned} \tag{2.7}$$

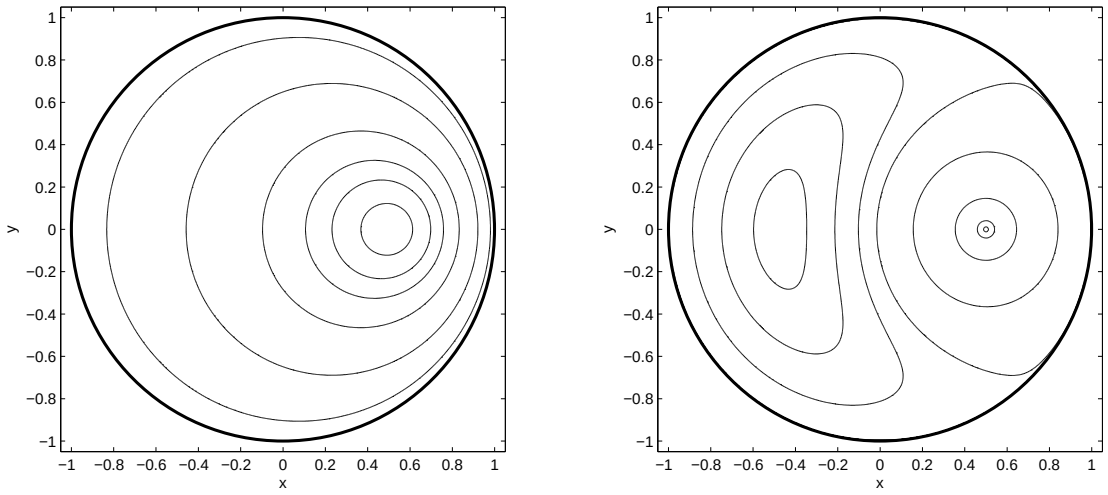
All the calculations are done in Cartesian coordinates, therefore following relations are used to compute the velocities in the directions of these coordinates, which are valid for all quadrants.

$$\begin{aligned} u_x(r, \theta, r_v, \theta_v(t)) &= u_r\cos(\theta) - u_\theta\sin(\theta) \\ u_y(r, \theta, r_v, \theta_v(t)) &= u_r\sin(\theta) + u_\theta\cos(\theta) \end{aligned} \tag{2.8}$$

The calculations can be done using a rotating frame, where the vortex is moving through the cylinder like in Fig. 2.1. The flow field of the gas becomes time-independent when the vortex is kept at one point and the cylinder is rotated. This will be named as the co-rotating frame. It can be realised by introducing $\phi = \theta - \theta_v(t)$. The stream function for the co-rotating frame becomes:

$$\hat{\Psi}(r, \phi) = \Psi(r, \phi + \theta) + \frac{1}{2} r^2 \dot{\theta}_v \tag{2.9}$$

For this case the velocities can again be determined using Eq. (2.6). The flow field of the co-rotating frame gives better insight in the behaviour of the flow. The contour lines of the stream functions for both frames are shown in Figure 2.2. In the rotating frame the core of the vortex rotates around the centre of the cylinder.




(a) Rotating frame $t = 0$ (b) Co-rotating frame

**Figure 2.2. A comparison of the contour lines of the stream functions for the rotating frame at $t = 0$ (a) and corotating frame independent of time (b).**

In further analyses all calculations are done using a rotating frame, so where the point vortex is moving through the domain.

## 2.2 FINITE-TIME LYAPUNOV EXPONENT

For the Lagrangian coherent structures it is needed to compute finite-time Lyapunov exponents (FTLE). With an FTLE calculated for each grid point the structures can be plotted. The FTLE is a scalar $\sigma_{t_0}^T(\mathbf{x})$ which represents the amount of stretching of the fluid at a

location in the domain over a certain time interval. The maxima show the repelling or attracting barriers for respectively a forward or backward integration time.

The FTLE can be derived by considering two neighbouring particles in a domain. A particle at $\mathbf{x}(t_0)\epsilon D$ is affected by the flow and therefore gets a new location after time $T$.

$$\mathbf{x} \mapsto \phi_{t_0}^{t_0+T}(\mathbf{x}) \tag{2.10}$$

The neighbouring point will behave similar as $\mathbf{x}(t_0)$ when it is affected by the same flow. When the time interval is increased, the distance between the two points will almost certainly change. The point close to $\mathbf{x}$ can be described as $\mathbf{y} = \mathbf{x} + \delta\mathbf{x}(t_0)$, where $\delta\mathbf{x}(t_0)$ is a infinitesimal distance. After a time interval $T$, the distance between the two points becomes:

$$\delta\mathbf{x}(t_0+T) = \phi_{t_0}^{t_0+T}(\mathbf{y}) - \phi_{t_0}^{t_0+T}(\mathbf{x}) = \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}\delta\mathbf{x}(t_0) + \mathcal{O}(\|\delta\mathbf{x}(t_0)\|^2) \tag{2.11}$$

Because the initial perturbation was infinitesimal, the higher order term $\mathcal{O}(\|\delta\mathbf{x}(t_0)\|^2)$ can be dropped. The perturbation then becomes:

$$\|\delta\mathbf{x}(t_0+T)\| = \sqrt{\left\langle \delta\mathbf{x}(t_0), \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}^* \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}\delta\mathbf{x}(t_0) \right\rangle} \tag{2.12}$$

Where $M^*$ indicates it is the adjoint matrix $M$. Here a finite-time version of the Cauchy-Green deformation tensor can be recognized:

$$\Delta = \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}^* \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}} \tag{2.13}$$

The maximum stretching, between the points $\mathbf{x}$ and $\mathbf{y}$, occurs when $\delta\mathbf{x}(t_0)$ is chosen such that it is aligned with the eigenvector of the maximum eigenvalue of $\Delta$.

$$\max_{\delta\mathbf{x}(t_0)} \|\delta\mathbf{x}(t_0+T) = \sqrt{\langle\overline{\delta\mathbf{x}}(t_0), \lambda_{max}(\Delta)\overline{\delta\mathbf{x}}(t_0)\rangle} = \sqrt{\lambda_{max}}\|\overline{\delta\mathbf{x}}(t_0)\| \tag{2.14}$$

where $\overline{\delta\mathbf{x}}(t_0)$ is aligned with the eigenvector associated with $\lambda_{max}(\Delta)$. If $\sigma_{t_0}^T(\mathbf{x})$ is now defined as:

$$\sigma_{t_0}^T(\mathbf{x}) = \frac{1}{|T|}\ln\sqrt{\lambda_{max}(\Delta)} \tag{2.15}$$

Eq. (2.14) can now be rewritten using Eq. (2.15) as:

$$\max_{\delta\mathbf{x}(t_0)} \|\delta\mathbf{x}(t_0+T) = e^{\sigma_{t_0}^T(\mathbf{x})|T|}\|\overline{\delta\mathbf{x}}(t_0)\| \tag{2.16}$$

Eq. (2.15) is representing the maximum finite-time Lyapunov exponent at point $\mathbf{x}\epsilon D$ at time $t_0$. The absolute value is taken for the integration time, because it should able to compute both the backward and forward FTLE. Eq. (2.15) can be be further rewritten using:

$$\|M\|_2 = \sqrt{\lambda_{max}(M^*M)} \tag{2.17}$$

Then Eq. (2.15) becomes:

$$\sigma_{t_0}^{T}(\mathbf{x}) = \frac{1}{|T|} \ln \left\| \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}} \right\|_2 \tag{2.18}$$

To compute the FTLE it is necessary to have the locations of traced particles at initial state $\mathbf{x}(t_0)$ and after the integration time $\mathbf{x}(t_0 + T)$. With the positions, the gradient of the flow map can be determined. This gradient is given by:

$$\left. \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}} \right|_{\mathbf{x}_{i,j}} = \begin{bmatrix} \frac{x_{i+1,j}(t_0+T) - x_{i-1,j}(t_0+T)}{x_{i+1,j}(t_0) - x_{i-1,j}(t_0)} & \frac{x_{i,j+1}(t_0+T) - x_{i,j-1}(t_0+T)}{y_{i,j+1}(t_0) - y_{i,j-1}(t_0)} \\ \frac{y_{i+1,j}(t_0+T) - y_{i-1,j}(t_0+T)}{x_{i+1,j}(t_0) - x_{i-1,j}(t_0)} & \frac{y_{i,j+1}(t_0+T) - y_{i,j-1}(t_0+T)}{y_{i,j+1}(t_0) - y_{i,j-1}(t_0)} \end{bmatrix} \tag{2.19}$$

The points in Eq. (2.19) correspond with the points shown in Figure 2.3.

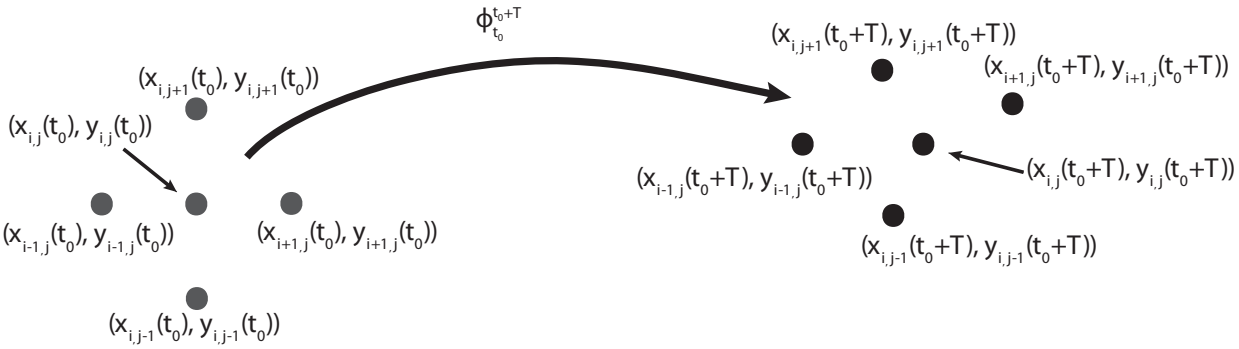



**Figure 2.3. Flowmap used for computing the FTLE.**

For the flow field of a double-gyre, for which the streamlines are shown in Figure 2.4, it is obvious that the particles will not cross the barrier between the two gyres. Therefore it is expected to see a barrier here. The forward FTLE field of this example is shown in Figure 2.5 and meets this expectation.

## 2.3 Particle Tracking

To compute the FTLE it is necessary to have the locations of the particles at initial state $\mathbf{x}(t_0)$ and after the integration time $\mathbf{x}(t_0 + T)$. To do this, different methods are used for the passive tracers and inertial particles.

### 2.3.1 Passive Tracers

A grid of passive tracers is released in the flow. At every point the velocities of the particles at $\mathbf{x}(t)$ can be determined using the analytical solution. The equation of motion Eq. (2.20) for the passive tracers is solved using the fourth order Runge-Kutta method.

$$\frac{d\mathbf{x}}{dt} = \mathbf{u} \tag{2.20}$$

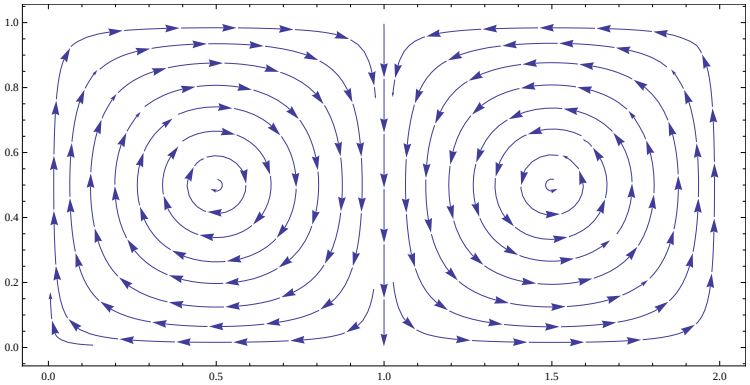



**Figure 2.4. Streamlines of a double gyre**

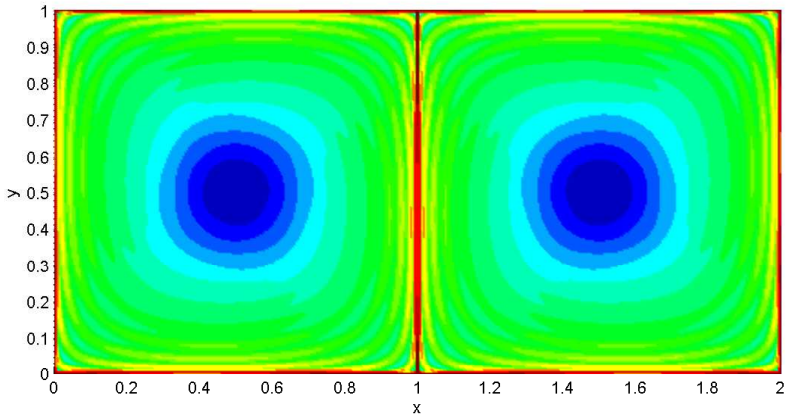



**Figure 2.5. Computed forward FTLE field for a double-gyre.**

$$\begin{aligned}
\mathbf{k_1} &= \Delta t \mathbf{u}(t, \mathbf{x}) \\
\mathbf{k_2} &= \Delta t \mathbf{u}\left(t + \frac{1}{2}\Delta t, \mathbf{x} + \frac{1}{2}\mathbf{k_1}\right) \\
\mathbf{k_3} &= \Delta t \mathbf{u}\left(t + \frac{1}{2}\Delta t, \mathbf{x} + \frac{1}{2}\mathbf{k_2}\right) \\
\mathbf{k_4} &= \Delta t \mathbf{u}(t + \Delta t, \mathbf{x} + \mathbf{k_3})
\end{aligned} \tag{2.21}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}) \tag{2.22}$$

Using Eq. (2.22) the next position can be computed for $t + \Delta t$ for each grid point. This method does not need previous steps, so it is immediately fourth order accurate. This is important due to the high vorticity near the centre of the vortex.

### 2.3.2 Inertial Particles

For the inertial particles it is much faster to use the three steps Adams-Bashforth method. For the Runge-Kutta method, it would be necessary to interpolate four times the velocities to the different points that are used because there is no analytical solution for the velocity. The initial velocities of the particles are the velocities of the gas at their positions. The inertial particles are assumed to only experience Stokes drag, which is only valid for low Reynolds numbers. The reduced equations Eq. (2.23) of motion that have to be solved are:

$$\begin{aligned} \frac{d\mathbf{x}_p}{dt} &= \mathbf{u}_p \\ \frac{d\mathbf{u}_p}{dt} &= \frac{1}{St}(\mathbf{u}_g - \mathbf{u}_p) \end{aligned} \tag{2.23}$$

where $\mathbf{x}_p$ is the position of the particle, $St$ is the Stokes number defined as $St = \frac{\tau_p \Gamma}{R^2}$, $\mathbf{u}_g$ the velocity of the gas and $\mathbf{u}_p$ the velocity of a inertial particle. Both equations are approximated by a three steps Adams-Bashforth method, which is given by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\Delta t}{12}(23\mathbf{u}_n - 16\mathbf{u}_{n-1} + 5\mathbf{u}_{n-2}) \tag{2.24}$$

As can be seen, this method needs two previous values to compute the next one. For the first step and second step the Euler method Eq. (2.25) and the two step Adams-Bashforth method Eq. (2.26) are used respectively.

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \mathbf{u}_n \tag{2.25}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{3}{2}\Delta t \mathbf{u}_n - \frac{1}{2}\Delta t \mathbf{u}_{n-1} \tag{2.26}$$

## 2.4 Global Attractivity of Inertial Particles

Inertial particles are not always affected by the FTLE. If the Stokes number is above a certain value, the inertial particles are not affected by the flow. A criteria is developed by Sapsis and Haller [8]. The condition can be tested for each grid point. With this condition it is now possible to make a distinction between between areas that are affected by the flow and areas that are not affected by the flow. They used the Maxey-Riley equations for the dynamics of the particles. They have rewritten the equations of motions to:

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{u}_p \\ \frac{d}{dt}(\mathbf{u}_p - \mathbf{u}_g) &= -(\nabla \mathbf{u}_g + \mu \mathbf{I})(\mathbf{u}_p - \mathbf{u}_g) \end{aligned} \tag{2.27}$$

With applying a change of coordinates $\mathbf{z} = \mathbf{u}_p - \mathbf{u}_g$, the following system is obtained:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{z} + \mathbf{u}_g \\ \dot{\mathbf{z}} &= -(\nabla \mathbf{u}_g + \mu \mathbf{I})\mathbf{z} \end{aligned} \tag{2.28}$$

with $\mu = \frac{2}{3St}$

If it is assumed that $[\mathbf{x}(t), \mathbf{z}(t)]$ is a solution, this can be substituted in Eq. (2.28). When the $\mathbf{z}$ component is multiplied by $\mathbf{z}(t)$ this will result in:

$$\begin{aligned} \frac{1}{2}\frac{d}{dt}|\mathbf{z}| &= -\langle \mathbf{z}, \{\nabla \mathbf{u}[\mathbf{x}(t), t] + \mu \mathbf{I}\}\mathbf{z}\rangle \\ &= \langle \mathbf{z}, [-\mathbf{S}(\mathbf{x}(t), t) - \mu \mathbf{I}]\mathbf{z}\rangle \\ &\leq \lambda_{max}[-\mathbf{S}(\mathbf{x}(t), t) - \mu \mathbf{I}]|\mathbf{z}|^2 \end{aligned} \tag{2.29}$$

Where $\lambda_{max}(T)$ is the maximum eigenvalue of the tensor $T$. Also here the rate-of-strain tensor is introduced:

$$\mathbf{S} = \frac{1}{2}[\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \tag{2.30}$$

When Eq. (2.29) is integrated, this results in:

$$|\mathbf{z}(t)| \leq |\mathbf{z}(t_0)| e^{-\int_{t_0}^{t} \lambda_{min}[\mathbf{S}(\mathbf{x}(s),s)+\mu \mathbf{I}]ds} \tag{2.31}$$

The subspace is globally attracting if:

$$\lim_{t\to\infty} \int_{t_0}^{t} \lambda_{min}[\mathbf{I} + \epsilon \mathbf{S}(\mathbf{x}(s; x_0), s)]ds = \infty \tag{2.32}$$

for all $x_0$ in the domain and where $\epsilon = 1/\mu$. This is satisfied when:

$$\lambda_{min}[\mathbf{I} + \epsilon \mathbf{S}(\mathbf{x}, t)] \geq c > 0 \tag{2.33}$$

If this is solved for two dimensional incompressible flow it is found to be:

$$\lambda^2 - 2\lambda + (1 + \epsilon^2 \det \mathbf{S}(\mathbf{x}, t)) = 0 \tag{2.34}$$

Therefore the two dimensional condition becomes:

$$\begin{aligned} \lambda_{min} &= 1 - \epsilon\sqrt{-\det \mathbf{S}(\mathbf{x}, t)} > 0 \\ &= \mu - \sqrt{|\det \mathbf{S}(\mathbf{x}, t)|} > 0 \end{aligned} \tag{2.35}$$

So when this criteria is satisfied on a location, the inertial particles are affected by the flow at this location.

## 2.5 DRIFT FLUX MODEL

For low Stokes numbers it is possible to derive an algebraic formula for the velocity of a particle. The slip (relative) velocity is defined as:

$$\mathbf{u}_{slip} = \mathbf{u}_p - \mathbf{u}_g \tag{2.36}$$

The velocity of the gas is already described, so to determine $\mathbf{u}_p$ it it necessary to calculate the slip velocity. Manninen *et al.* [6] derived an algebraic expression for the slip velocity. A balance equation can be derived from using the momentum equation for the inertial particles and the mixture. This results in the following equation:

$$\mathbf{F}_d = V_p(\rho_p - \rho_m)\left[\mathbf{g} - (\mathbf{u}_m \cdot \nabla)\mathbf{u}_m - \frac{d\mathbf{u}_m}{dt}\right] \tag{2.37}$$

Here $V_p$ is the volume of a inertial particle and $\rho_p$ and $\rho_m$ are the densities of respectively the inertial particles and the mixture. The formula for the Stokes drag ($\mathbf{F}_d$) is:

$$\mathbf{F}_d = 3\pi d_p \mu_g \mathbf{u}_{slip} \tag{2.38}$$

The gravity is neglected and the density of the inertial particles is assumed to be much higher than the density of the gas. Furthermore the velocity of the mixture is assumed to be the same as the velocity of the gas, which is a reasonable approximation for low Stokes numbers. The flow field of the gas particles is steady, so the time derivative can be dropped. This results in:

$$\begin{aligned} 3\pi d_p \mu_g \mathbf{u}_{slip} &= V_p(\rho_p - \rho_m)(-(\mathbf{u}_g \cdot \nabla)\mathbf{u}_g) \\ \mathbf{u}_{slip} &= -\frac{\rho_p {d_p}^2}{18\mu_g}(\mathbf{u}_g \cdot \nabla)\mathbf{u}_g \end{aligned} \tag{2.39}$$

The particle relaxation time for particles in the Stokes regime is Eq. (2.40).

$$\tau_p = \frac{\rho_p {d_p}^2}{18\mu_g} \tag{2.40}$$

The slip velocity can now be written as:

$$\mathbf{u}_{slip} = -St(\mathbf{u}_g \cdot \nabla)\mathbf{u}_g \tag{2.41}$$

The final algebraic formula for the velocity of the inertial particles therefore becomes:

$$\mathbf{u}_p = \mathbf{u}_g - St(\mathbf{u}_g \cdot \nabla)\mathbf{u}_g \tag{2.42}$$

# CHAPTER 3
# VALIDATION

To validate if the code is working properly, a few checks are performed to see if it behaves like it is expected to be. First the time step is determined, which will be used for all the simulations. The fluid particles are traced using different time steps and are compared to the results when a very small time step is used. To determine the FTLE, it is also necessary to choose an integration length. The code can be compared to an analytical computation of a FTLE for a simple case.

## 3.1 TIME STEP

In this section the time step is determined, which will be used for all the simulations. The fluid particles are traced using different time steps and afterwards compared to the results when a very small time step is used. These results should converge to a results of where a very small time step is used. The $L_1\ error$ is shown in Figure 3.1. This figure shows that the error is of fourth order as expected.

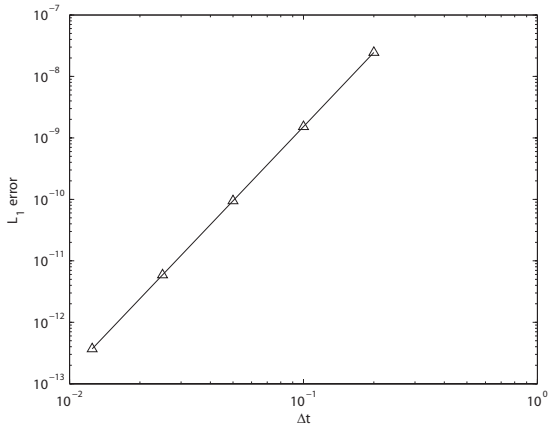



**Figure 3.1.** $L_1\ error$ **of the computed x-position for different time steps with the fourth order Runge-Kutta method.**

To see which time step should be used, the FTLE is compared for the same time steps that were used for the $L_1\ error$. It is easy to compare the results when a fixed y-position is used, then the FTLE is only depending on the x-position. The FTLE for $y = 0.0025$ is shown in Figure 3.2. In this figure the area around the vortex is shown. In this area the velocities are

very high and therefore the time step should be very small to give converging results. The area around the vortex is not the region of interest, which will be explained in section 4.2.2. The time step is chosen in a way that the first ridge of the FTLE shows good similarity with the ridge of the FTLE with a very small time step. This ridge is important for the behaviour of the particles and should therefore be positioned at the right position. The time step that will be used is $\Delta t = 0.025$. To see how exact the FTLE is in the rest of the region the FTLE is also

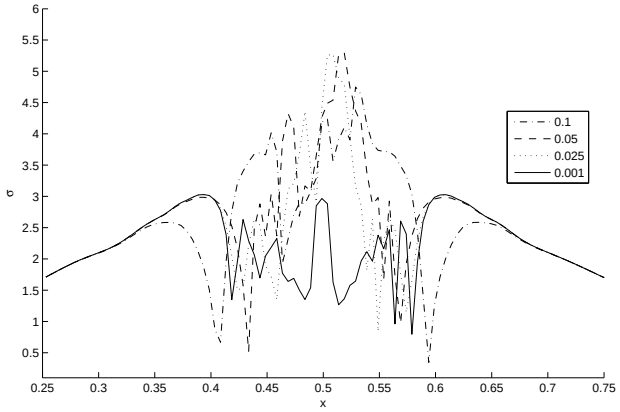



**Figure 3.2. FTLE for different time steps $L_1$ $error$ for $y = 0.0025$.**

compared for another y-position $y = 0.5038$. These results are shown in Figure 3.3. For the areas with lower velocities all the used time steps are sufficient.

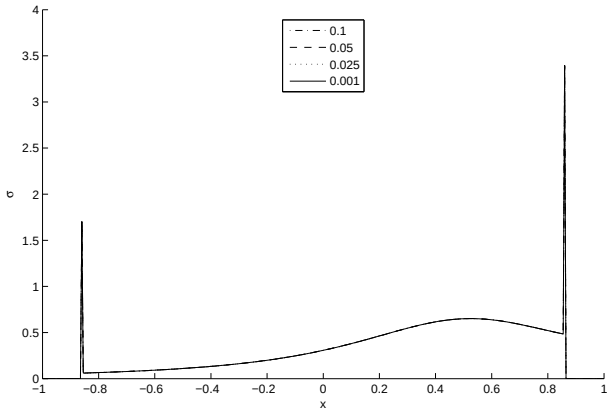



**Figure 3.3. FTLE for different time steps $L_1$ $error$ for $y = 0.5038$.**

## 3.2 PARTICLES IN DOMAIN

Due to the high velocities at the centre of the vortex, it is possible that particles leave the domain when the time step is too large. The gas particles should always stay in the domain. Particles that leave the domain are programmed to stick to the wall and will stay there. A row of particles is traced to see of they will stay in the domain or not. The results are shown in Figure 3.4. At the initial state, fifteen particles are traced in the domain. One particle is very close to the vortex and one particle is very close to the wall. After one time step there are only fourteen particles left in the domain. The particle closest to the vortex left the domain immediately. At $t = 95$ there are still fourteen particles in the domain. These particles keep making rounds around the centre of the vortex and are not pushed outwards.

## 3.3 INTEGRATION LENGTH

The integration length for the evaluation of the FTLE is very important. When the integration length is chosen very small, there are no structures to see. When the integration length is increased more ridges will be visible. To see if this statement is true for the code, the FTLE field is determined for various integration lengths which is shown in Figure 3.5. Longer integration times results in sharper ridges and therefore show more information.

## 3.4 ANALYTIC FTLE COMPARISON

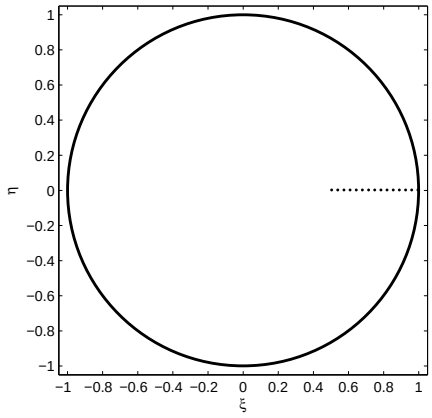
For very simple flow fields the FTLE can be determined analytically. When the Cauchy-Green deformation tensor can be described analytically, the FTLE can be computed analytically too. For the simple case of a strain flow, the flow field can be considered as:

$$\begin{aligned} \frac{dx}{dt} &= x \\ \frac{dy}{dt} &= -y \end{aligned} \tag{3.1}$$

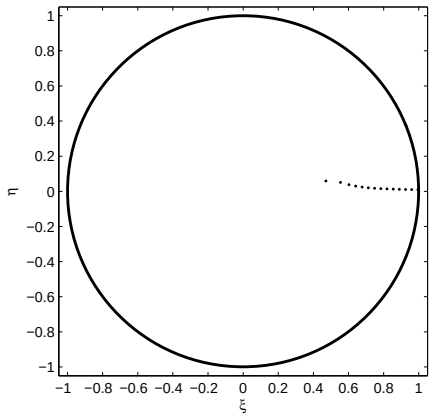
For this case the Cauchy-Green deformation tensor is:

$$\Delta = \begin{bmatrix} e^{2T} & 0 \\ 0 & e^{-2T} \end{bmatrix} \tag{3.2}$$

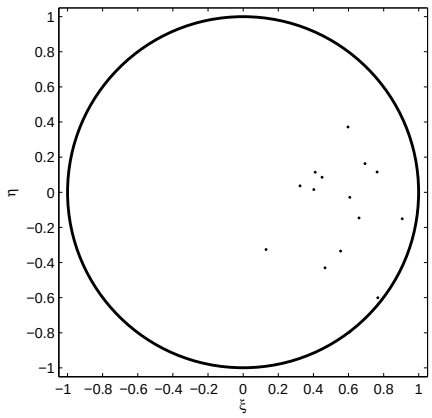
The analytic FTLE can now be calculated using Eq. 2.15. The $L_1\ error$ is now determined between the analytical solution and the solution computed by the code. The error is shown in Figure 3.6. Here it is shown again that the error is of fourth order as expected.




(a) $t = 0$




(b) $t = 0.025$




(c) $t = 95$

**Figure 3.4. Positions of particles in the domain at (a)** $t = 0$**, (b)** $t = 0.025$ **and (c)** $t = 95$**.**

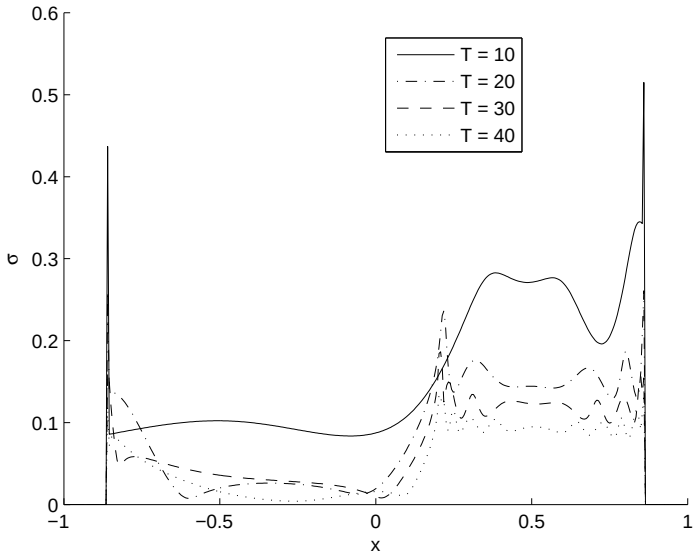



**Figure 3.5. FTLE for different integration lengths for $y = 0.5038$.**

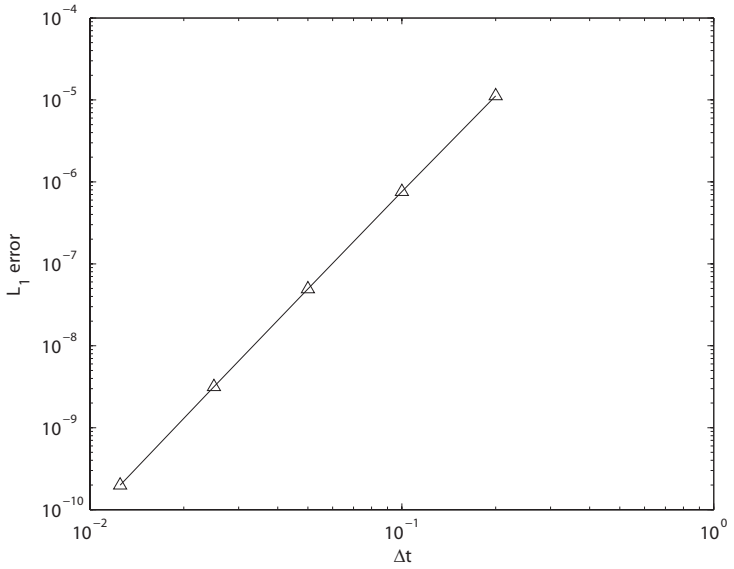



**Figure 3.6.** $L_1\ error$ **of the computed FTLE for different time steps with the fourth order Runge-Kutta method.**

# CHAPTER 4
# RESULTS

In the chapter the results will be shown for the vortex flow on a disk. First the accumulation of the inertial particles will be shown and their dependence on the Stokes number. With all the data, the FTLE field can be determined from the gas fluid particles, which also can be done for the inertial particles. These results are compared to the results of the drift flux model, which can also be used for the calculations of the inertial particles.

## 4.1 PARTICLE ACCUMULATION

The inertial particles will accumulate in a point for a range of Stokes numbers. For high Stokes numbers, all the inertial particles will move to the wall. When the Stokes number is very low, it will more behave like the gas. For the pictures it is more convenient to use the frame where the vortex stays at one point. That calculations are done in the rotating frame, but the pictures are rotated back using:

$$\begin{aligned} \xi &= r\cos(\theta - \theta_v(t)) \\ \eta &= r\sin(\theta - \theta_v(t)) \end{aligned} \tag{4.1}$$

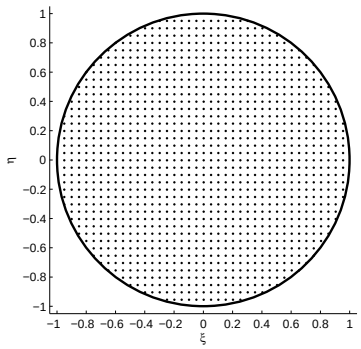
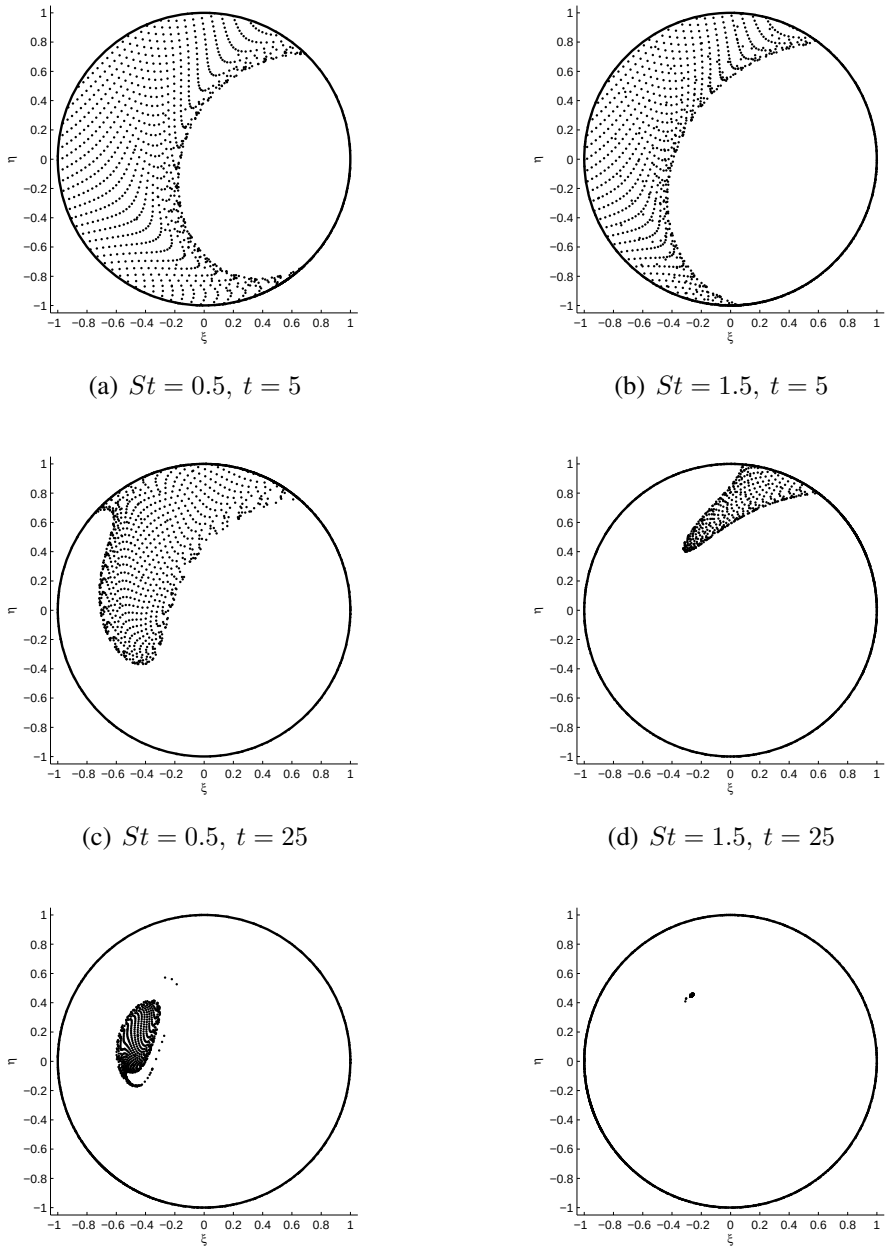



**Figure 4.1. Initial positions of the inertial particles $t = 0$.**

A square grid of $400 \times 400$ points is used for all the computations. Only the particles in the circular domain are tracked. The initial positions of the particles are shown in Figure 4.1. The positions of the particles for $St = 0.5$ and $St = 1.5$ at different times are shown in

Figure 4.2. As can be seen in the figures, the particles accumulate much faster when the Stokes number is increased. The location of accumulation is also depending on the Stokes number. The particles which do not accumulate are absorbed by the wall and will stay there. These results show good similarity with IJzermans and Hagmeijer [4].




(a) $St = 0.5,\ t = 5$

(b) $St = 1.5,\ t = 5$

(c) $St = 0.5,\ t = 25$

(d) $St = 1.5,\ t = 25$

(e) $St = 0.5,\ t = 100$

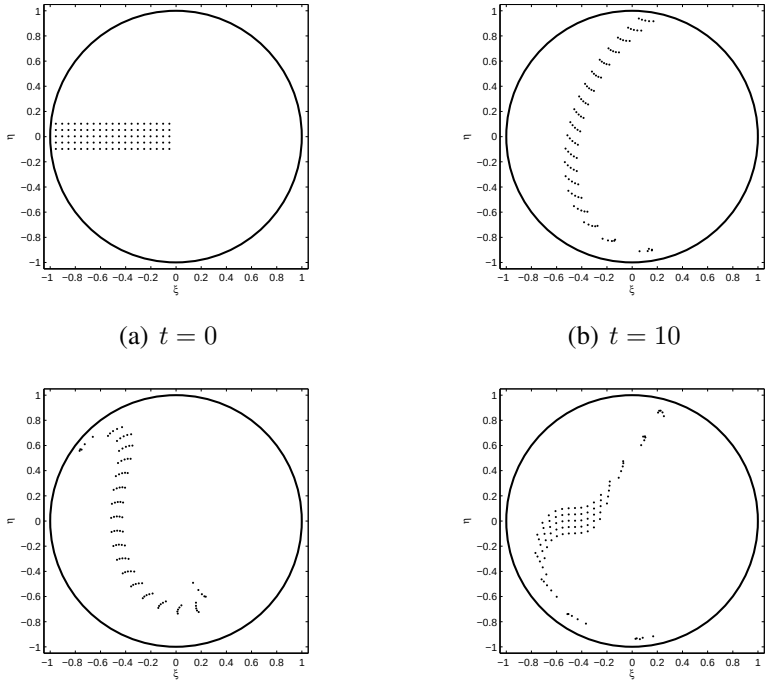
(f) $St = 1.5,\ t = 100$

**Figure 4.2. Accumulation of inertial particles $St = 0.5$ and $St = 1.5$ at $t = 5$, $t = 25$ and $t = 100$ in the co-rotating frame.**

## 4.2 LAGRANGIAN COHERENT STRUCTURES

### 4.2.1 Periodic FTLE

To calculate the FTLE field it is necessary to use a certain integration length. In this specific situation the FTLE is periodic with the time it needs for the particles to rotate one time around the centre. This can be illustrated by Figure 4.3. As can be seen from Figure 2.2(b) the particles in the left half of the cylinder will rotate around a centre. Near this centre the particles are stretched and squeezed in a periodically. At $t = 40.1$ these particles complete one lap. For the particles near to the wall this takes longer then the particles in the centre, so the tail keeps getting longer over time. The corresponding FTLE's for different integration lenghts are shown in Figure 4.4. As can be seen from Figure 4.3(c) and 4.3(d) the structure in the FTLE looks similar. This is the FTLE calculated after two laps around the accumulation point. The FTLE field looks periodically similar, but is rather different during one period. To get a better representation of the FTLE it will be averaged over a period. The averaging time that is used, is the time the when first particles have completed one lap around a centre. These averaging times for different Stokes numbers are shown in table 4.1 and were determined using MATLAB.



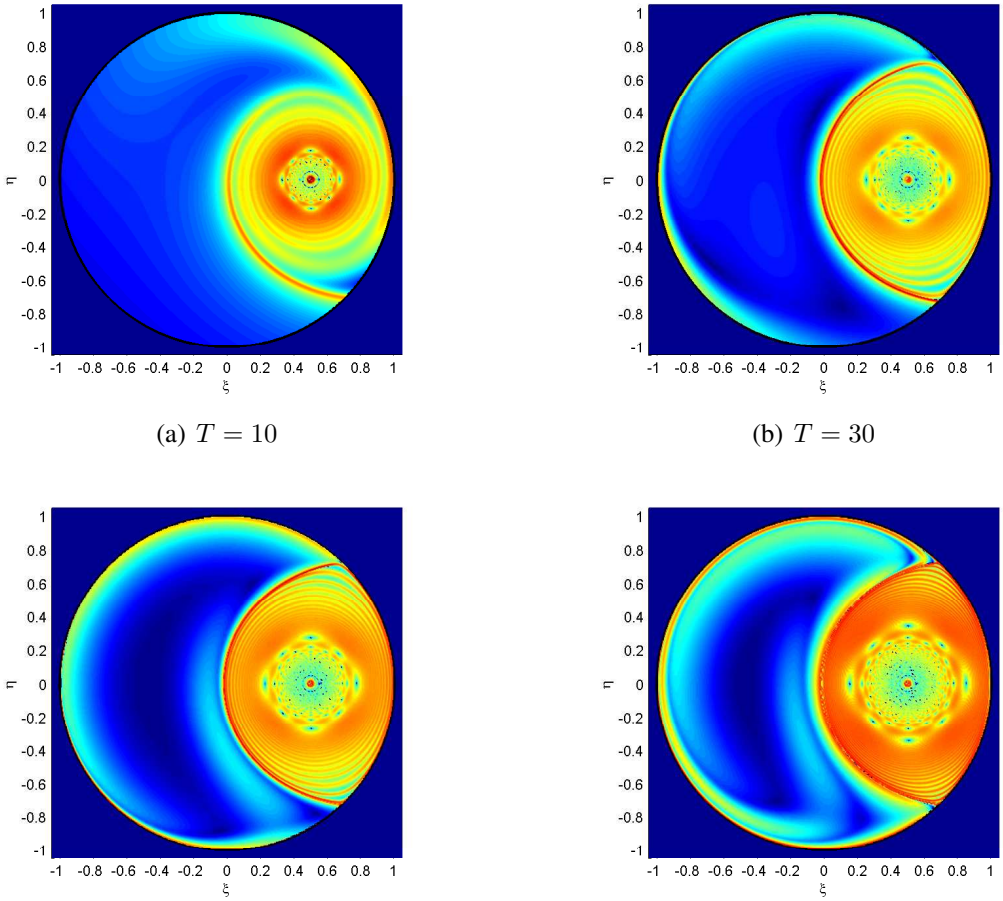

(a) $t = 0$ (b) $t = 10$

(c) $t = 30$ (d) $t = 40.1$

**Figure 4.3. Tracing of fluid particles in the left half of the domain at (a) $t = 0$, (b) $t = 10$, (c) $t = 30$ and (d) $T = 40.1$ in the co-rotating frame.**

**Table 4.1. Averaging times used to compute the averaged FTLE field.**

| Stokes number | Averaging time |
|---|---|
| - (Fluid) | 40.1 |
| 0.05 | 40.1 |
| 0.1 | 40.125 |
| 0.5 | 40.6 |
| 1.5 | 52.275 |



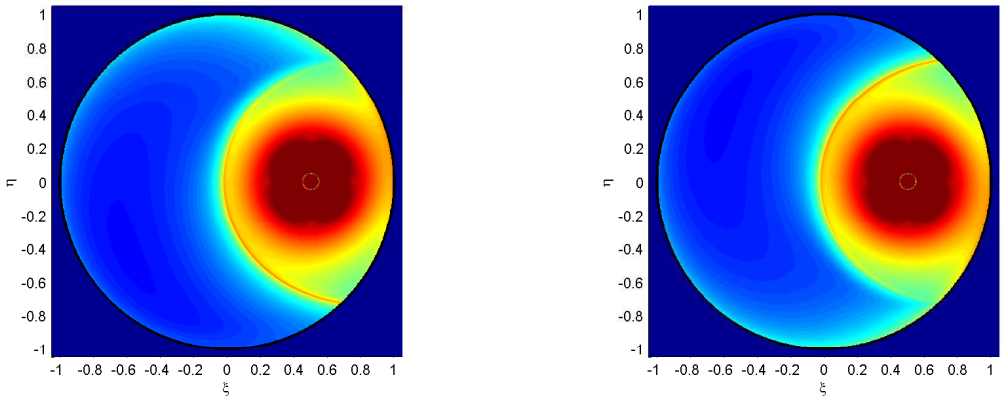

(a) $T = 10$ (b) $T = 30$

(c) $T = 40.1$ (d) $T = 80.2$

**Figure 4.4. Computed forward FTLE fields of fluid particles with different integration lengths (a) $T = 10$, (b) $T = 30$, (c) $T = 40.1$ and (d) $t = 80.2$ in the co-rotating frame.**

### 4.2.2 FTLE of Fluid Particles

The FTLE is calculated with an integration length of one period $T = 40.1$ and averaged over this time as described in section 4.2.1. This results in the FTLE field shown in Figure 4.5. In the left half of the cylinder the values are very low and therefore there are no real structures in this area where the accumulation point is. The accumulation point is a fixed point in the co-rotating frame and was calculated by IJzermans and Hagmeijer [4]. This point is located where $u_\theta = 0$. Because the flow is steady, the FTLE field is constant over time and therefore only has to be calculated once. When the FTLE is calculated at the initial time, this can be used further in time. When the backward FTLE is calculated, the vortex is rotating in the counter direction and the vortex itself is rotating around the centre of the cylinder in the counter direction. Then it is obvious to see the backward FTLE field is the mirrored image of the forward FTLE field.




(a) Forward FTLE (b) Backward FTLE

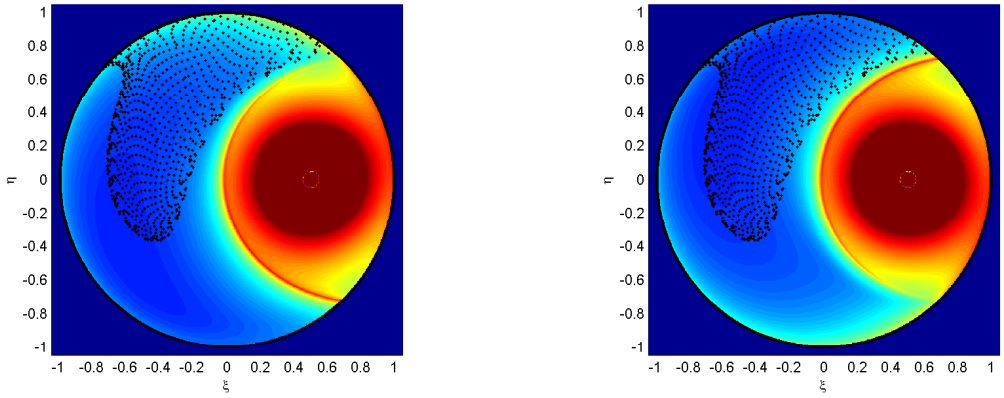
**Figure 4.5. Averaged FTLE fields of gas particles with an integration and averaging time of $T = 40.1$ in the co-rotating frame.**

In Figure 4.6 the FTLE field is shown and the inertial particles are plotted on top. The particles follow the shape of the structures shown in the FTLE field.

For the FTLE the criteria explained in section 2.4, can be plotted on top of the FTLE to distinguish areas which are affected by the FTLE. This is done in Figure 4.7. As can be seen from this figure, inertial particles near the core of the vortex are not affected by the FTLE field. For higher Stokes numbers the area that is not affected, becomes larger.

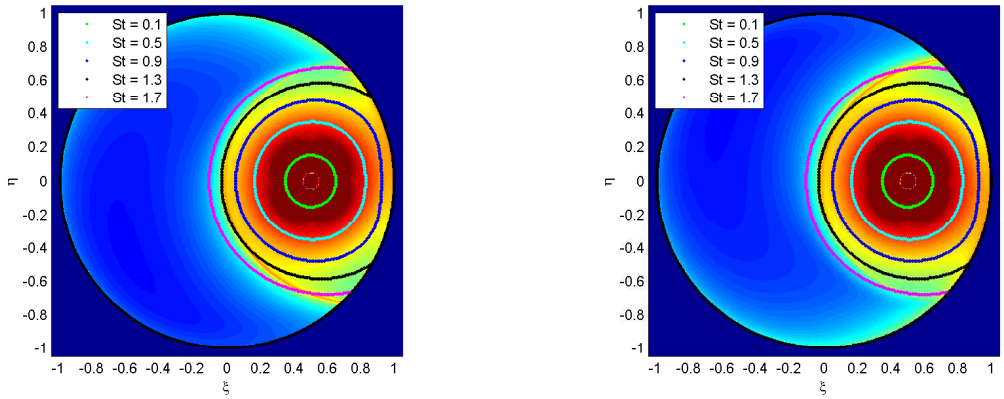
### 4.2.3 FTLE of Inertial Particles

The FTLE field can also be determined for the inertial particles. This is done similar to the gas particles, due to the behaviour of the inertial particles this will result in different




(a) Forward FTLE (b) Backward FTLE

**Figure 4.6. Averaged FTLE fields of gas particles with inertial particles $St = 0.5$ at $t = 25$.**

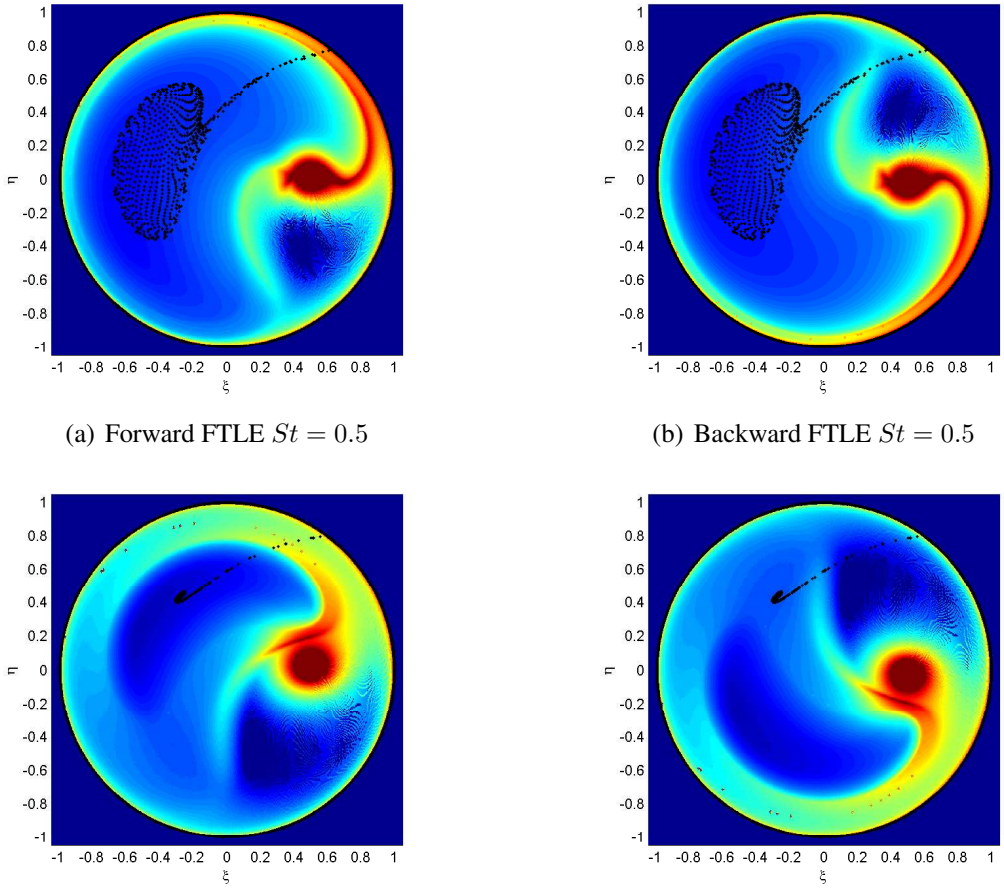



(a) Forward FTLE (b) Backward FTLE

**Figure 4.7. Criteria for the distinction of areas which are not affected by the FTLE fields.**

Lagrangian coherent structures. For two different Stokes numbers this is shown in Figure 4.8. The inertial particles at $t = 100$ are plotted on top of the FTLE fields. For the higher Stokes number the area with low FTLE values becomes smaller and the higher FTLE values, which were found near the wall, can now be found more inward the cylinder. The tail of the converging particles is cut by a saddle point. Particles will not cross this border, because this is a repeller. On one side the particles will move to the accumulation point and at the other side these particles will move to the wall.

The initial velocity of an inertial particle is the gas velocity at that location. To see what effect the initial velocity has on the results, the same calculations are done for the




(a) Forward FTLE $St = 0.5$

(b) Backward FTLE $St = 0.5$

(c) Forward FTLE $St = 1.5$

(d) Backward FTLE $St = 1.5$

**Figure 4.8. Forward and backward FTLE fields for two different Stokes numbers plotted with inertial particles at $t = 43.75$.**

inertial particles where the initial velocities are zero. The FTLE field for this situation is shown in Figure 4.9. This looks similar to the results of Figure 4.8(a) except for the area near the location of the vortex.

The positions of the inertial particles for the different initial velocities are compared in Figure 4.10. As can be seen from this figure, the particles will accumulate in the same point and the structure of the particles looks the same for both initial conditions.

## 4.3 Drift Flux Model

For low Stokes numbers it is possible to use an analytical flow field for the inertial particles. To show this, the calculations with the drift velocity are compared to the results of the Adams-Bashforth method. The formulas for these velocities can be found in Appendix B. When the drift velocity is used for the inertial particles, there appears a source at the position of the vortex. It is not possible to give a good representation of the backward FTLE field,

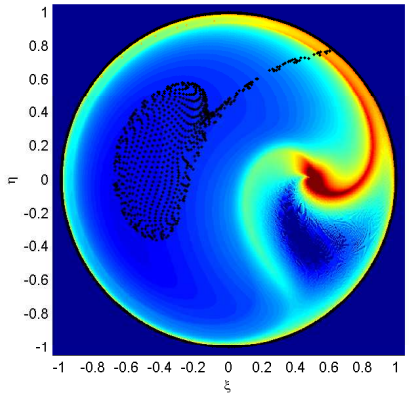
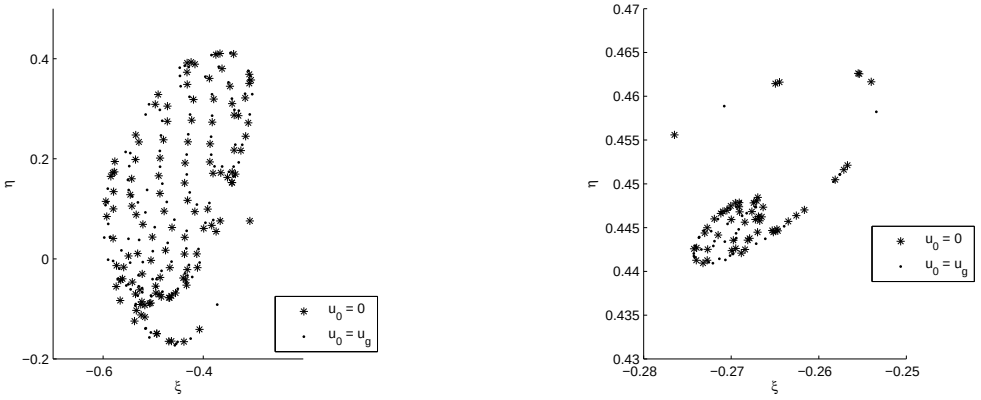



**Figure 4.9. Forward FTLE field for inertial particles with zero initial velocity.**

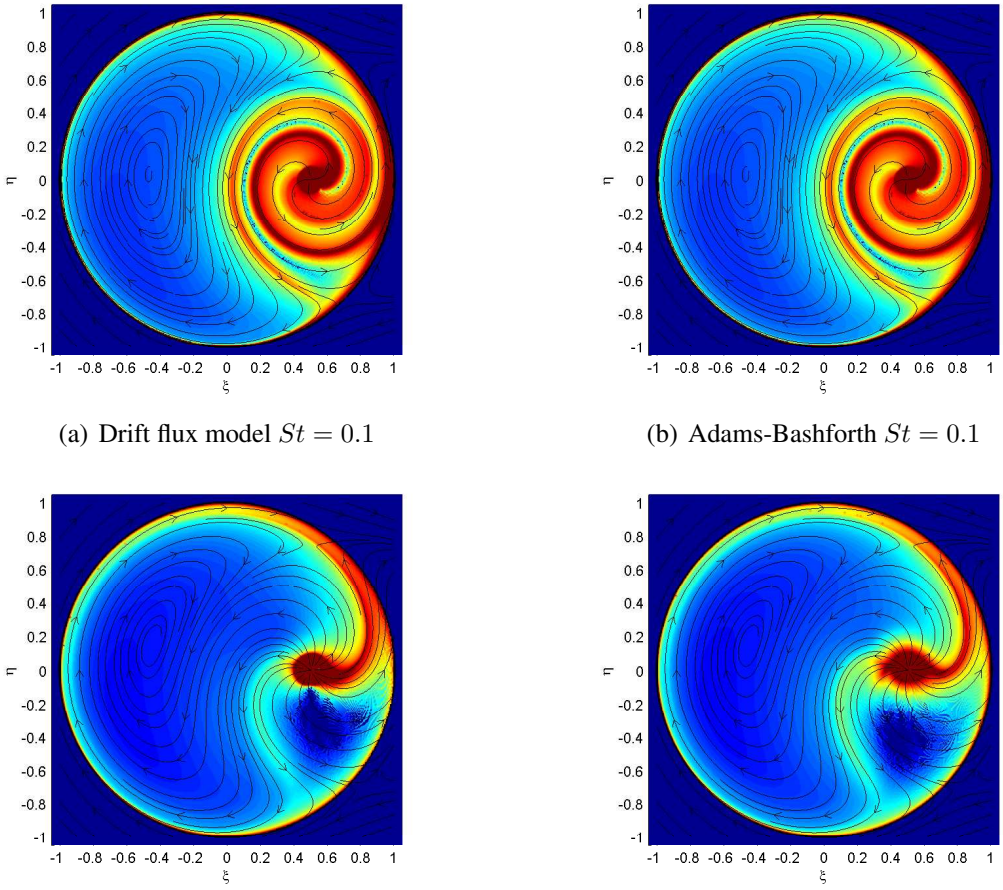



(a) $St = 0.5$ (b) $St = 1.5$

**Figure 4.10. Comparison for the positions of the inertial particles with different initial velocities at $t = 100$.**

therefore only the forward FTLE's are compared. For two different Stokes numbers both methods are compared in Figure 4.11. From the streamlines, two saddle points can be observed. For low Stokes numbers the FTLE field looks the same. For $St = 0.5$ the structure looks similar, but the line going down towards a saddle point does not match anymore. Ofcourse the FTLE field of the drift flux model matches perfectly with the streamlines of this model, but the results of the Adams-Bashforth method show no perfect match.

Particles which start in the region around the vortex will not cross the repellers. From Figure 4.11 it is easy to see which particles will immediately move to the wall and which will accumulate. To show the separatrix of the backward FTLE, it is possible to trace a particle which leaves at the saddle point. This is shown in Figure 4.12.




(a) Drift flux model $St = 0.1$

(b) Adams-Bashforth $St = 0.1$

(c) Drift flux model $St = 0.5$

(d) Adams-Bashforth $St = 0.5$

**Figure 4.11. Comparing the FTLE fields calculated by the drift flux model and Adams-Bashforth with the streamlines of the drift flux model.**

To see how well the inertial particles follow the streamlines the drift flux model, two particles are traced using both methods. This is done for different Stokes numbers and is shown in Figure 4.13. For low Stokes numbers $St << 1$ both paths are similar and therefore should give the same FTLE fields. When the Stokes number is increased the paths are different, but the accumulation point is still similar. When the Stokes number is further increased the paths show less similarity and it is possible the streamline is converging and the inertial particle is absorbed by the wall. The FTLE fields should not be similar anymore for higher Stokes numbers, which was already seen in the earlier comparison.

Furthermore it is possible to take a look how the drift velocity predicts the accumulation point. In Figure 4.14 is shown where the accumulation point will be according to the drift velocity and the inertial particles which were tracked by the Adams-Bashforth method. As can be seen from the figure the accumulation point of the inertial particles and the

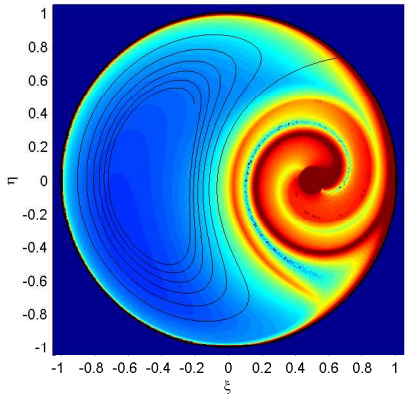
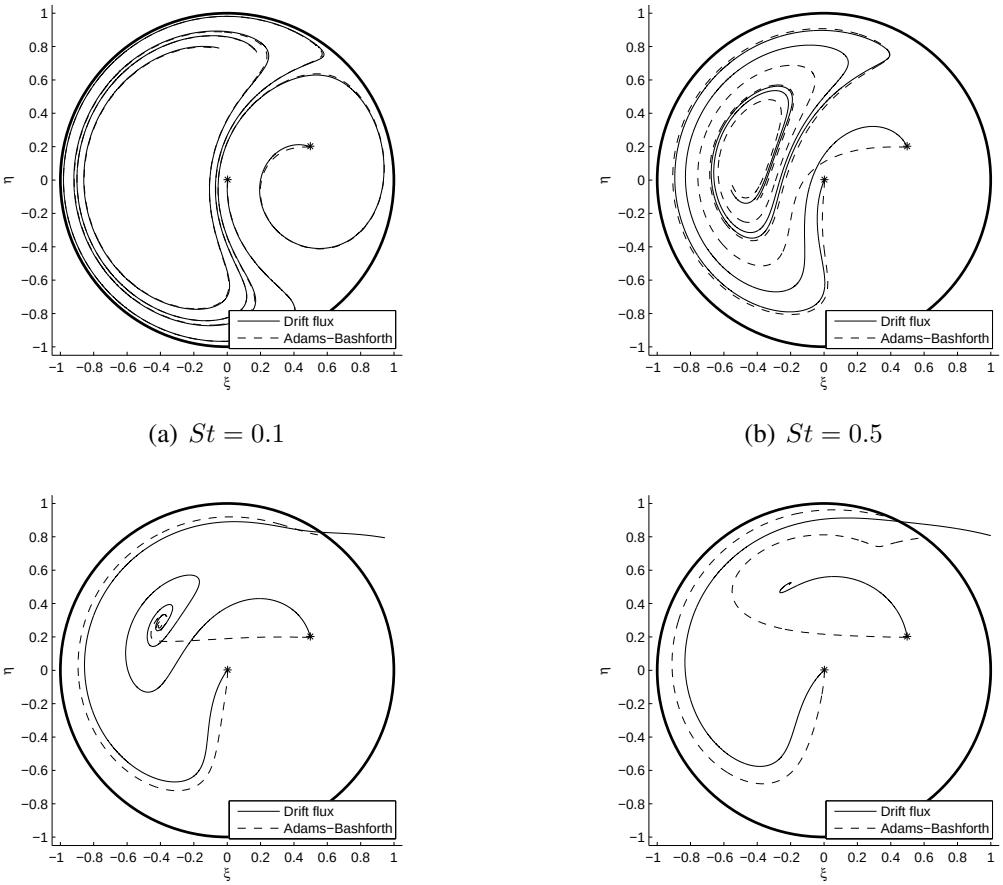



**Figure 4.12. Streamline of a particle which was traced from the saddle point, representing a separatrix on the forward FTLE field.**

one shown by the streamlines show good agreement for low Stokes numbers. When the Stokes number is increased, the difference between the two becomes bigger. According to the streamlines, the particles will not accumulate when $St > 1.3$. As can be seen from the inertial particles, they will not accumulate anymore when $St > 1.7$.



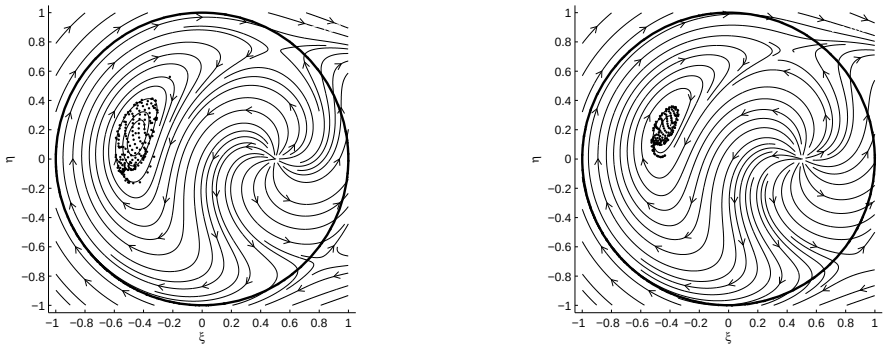

(a) $St = 0.1$

(b) $St = 0.5$

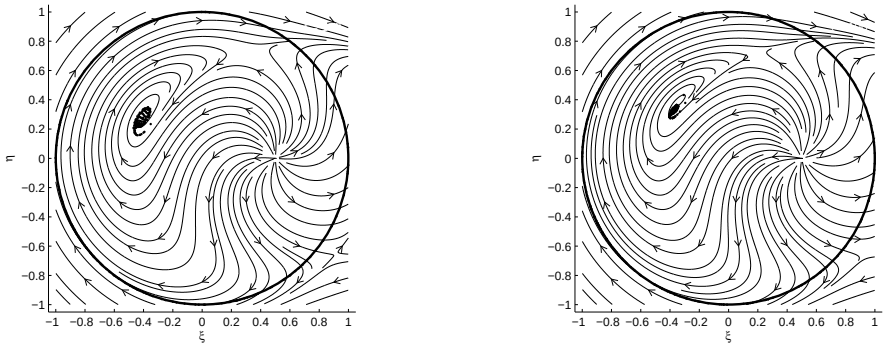
(c) $St = 0.9$

(d) $St = 1.3$

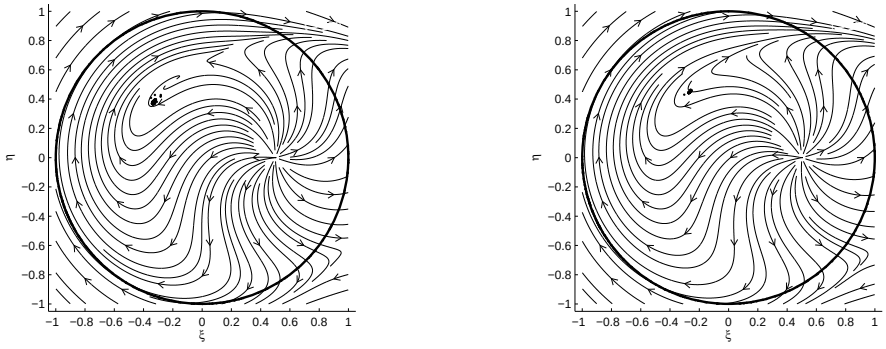
**Figure 4.13. Comparing the results of tracing two particles by using the drift flux model and Adams-Bashforth for different Stokes numbers.**



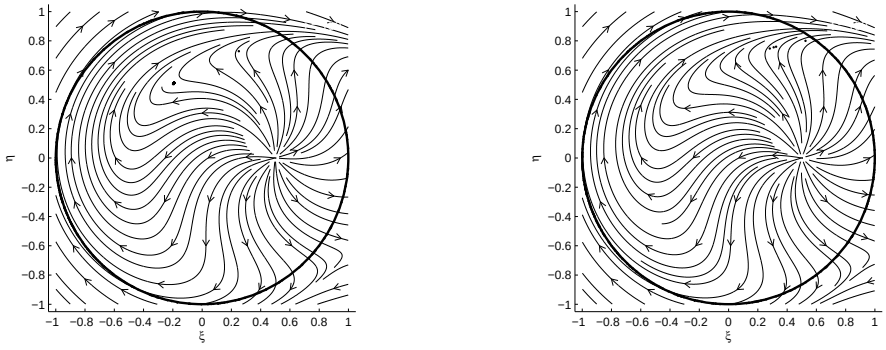

(a) $St = 0.5$ (b) $St = 0.7$




(c) $St = 0.9$ (d) $St = 1.1$




(e) $St = 1.3$ (f) $St = 1.5$




(g) $St = 1.7$ (h) $St = 1.9$

**Figure 4.14. Comparing the position of accumulation point according to Adams-Bashforth and the drift flux model for different Stokes numbers. All at** $t = 100$**.**

# CHAPTER 5
# CONCLUSIONS AND RECOMMENDATIONS

In this chapter the final conclusions will be drawn and some recommendations are done for possible future work.

Due to the stretching and squeezing of the particles in the converging area of the cylinder, the FTLE fields of this problem are periodic. The particles keep spinning around a point, where the particles will accumulate after some time. The resulting FTLE field is strongly depending on the integration length. When an integration length is used, it is possible to see some structures but is not a good representation for the FTLE field over time. Therefore the FTLE is averaged over time to get a better representation. The results are now depending on the time over which it is averaged, but has much less influence on the final results.

The Lagrangian Coherent Structures of the vortex flow on a disk do not show a attractor to the observed accumulation point. They do show an attractor and repeller between two areas. The inertial particles will move along these borders as expected by the LCS. Because there is no real attractor or repeller towards the accumulation point, it is not possible to predict the accumulation point with the FTLE fields.

With the LCS of the inertial particles it is possible to see where the saddle points are located. It is also possible to see which particles will move to the wall and which particles will move to the accumulation point. The saddle point moves further towards the centre of the cylinder. When the saddle point moves inwards, there will be more particles moving between the the wall and this separatrix. This means more particles will move to the wall, when the Stokes number is increased. The initial velocities have no significant influence on the results. The particles will behave similar and accumulate to the same point.

The drift flux model shows for $St << 1$ good similarity with the results of the Adams-Bashforth method. For higher Stokes numbers the path of the particles become different, but the accumulation point stays approximately at the same location. When the Stokes number is further increased $St > 1$, the paths of the particles are different and the accumulation point predicted by the drift flux model is different from the accumulation point calculated with Adams-Bashforth. Therefore the drift flux model can be used for $St << 1$ to calculate the FTLE fields and the accumulation point, but when the Stokes number is increased the FTLE fields do not match. For $St < 1$, the drift flux model still gives a good representation of the accumulation point.

In order to fully understand the accumulation of the particles more research is needed. The model that is used is very simplified and the use of a two-way coupled model could be interesting. The flow will not be steady anymore and therefore the FTLE will not be time-independent anymore. The code can also be used for other cases to compute FTLE's. Any other analytical flow field can be inserted into the code and this will solve the equations of motion and compute the corresponding FTLE fields.

# BIBLIOGRAPHY

[1] G. HALLER, *Finding finite-time invariant manifolds in two-dimensional velocity fields*, Chaos, 10 (2000), pp. 99–108.

[2] G. HALLER AND T. SAPSIS, *Where do inertial particles go in fluid flows?*, Physica D, 237 (2008), pp. 575–583.

[3] J. HARDIN, *The velocity field induced by a helical vortex filament*, Physics of Fluids, 25 (1982), p. 1949.

[4] R. IJZERMANS AND R. HAGMEIJER, *Accumulation of heavy particles in n-vortex flow on a disk*, Physics of Fluids, 18 (2006), p. 063601.

[5] H. LAMB, *Hydrodynamics*, Cambridge University Press, Cambridge, 1932.

[6] M. MANNINEN, V. TAIVASSALO, AND S. KALLIO, *On the mixture model of multiphase flow*, VTT publications, 288 (1996).

[7] L. MILNE-THOMSON, *Theoretical Hydrodynamics*, The Macmillan Company, New York, 1968.

[8] T. SAPSIS AND G. HALLER, *Instabilities in the dynamics of neutrally buoyant particles*, Physics of Fluids, 18 (2008), p. 017102.

[9] S. SHADDEN, F. LEKIEN, AND J. MARSDEN, *Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional flows*, Physica D, 212 (2005), pp. 271–304.

# APPENDIX
# MATLAB SCRIPT

# MATLAB SCRIPT

## A.1 GAS PARTICLES

```
clear all; clc; close all

%--input--
nx = 400;           %number of gridpoints
ny = 400;

xstart = -1;        %domain
xlast  =  1;
ystart =  1;
ylast  = -1;

dt = 0.025;         %integration time step

write = 0;          %1 for writing datafiles

r_j = 0.5;          %radius of the point vortex

t_start = 0;        %initial time time
t_end = 100;        %end time of the run

d = 1;              %direction of ftle, forward = 1, backward = -1

if d == 1
    direction = 'forward';
elseif d == -1
    direction = 'backward';
end

name = ['LCS_' direction '_gas_0025_'];     %name of datafile

corotating = 0;     %setting frame, 1 = corotating, 0 = rotating

%----------

xcoor = linspace(xstart, xlast, nx);        %vectors for meshgrid
ycoor = linspace(ystart, ylast, ny);
```

```
37
38 [X0,Y0] = meshgrid(xcoor, ycoor);       %initial positions of particles
39
40 %defining matrices
41 Up = zeros(nx,ny);
42 Vp = zeros(nx,ny);
43 Up1 = zeros(nx,ny);
44 Vp1 = zeros(nx,ny);
45 Xp1 = zeros(nx,ny);
46 Yp1 = zeros(nx,ny);
47
48 Xg1 = zeros(nx,ny);
49 Yg1 = zeros(nx,ny);
50 Xg2 = zeros(nx,ny);
51 Yg2 = zeros(nx,ny);
52 Ug1 = zeros(nx,ny);
53 Vg1 = zeros(nx,ny);
54 Ug2 = zeros(nx,ny);
55 Vg2 = zeros(nx,ny);
56
57 k1x = zeros(nx,ny);
58 k1y = zeros(nx,ny);
59 k2x = zeros(nx,ny);
60 k2y = zeros(nx,ny);
61 k3x = zeros(nx,ny);
62 k3y = zeros(nx,ny);
63 k4x = zeros(nx,ny);
64 k4y = zeros(nx,ny);
65
66 sigma = zeros(nx,ny);
67
68 domain = zeros(nx,ny);
69 %boundaryp = zeros(nx,ny);
70 crit = zeros(nx,ny);
71
72
73 Z =zeros(nx,ny);
74
75 for i = 1:ny
76     for j =1:nx
77         if sqrt(X0(i,j)^2+Y0(i,j)^2) ≤ 1
78             domain(i,j) = 1;        %the particles in the domain ...
                  (circle with r=1) will be traced
79         end
```

```
    end
end

theta_j = @(t) (1/(2*pi))*(1/(1-r_j^2))*t;      %angular position of ...
   point vortex
theta_jt = (1/(2*pi))*(1/(1-r_j^2));            %angular velocity of ...
   point vortex

u_r_func = @(r,theta,t) (1./r).*((-1./(4.*pi).*(2.*r.*r_j.* ...
    sin(theta-theta_j(t)))./(r.^2+r_j.^2-2.*r.*r_j.* ...
    cos(theta-theta_j(t))))-(-1./(4.*pi).*(2.*r.*r_j.^-1.* ...
    sin(theta-theta_j(t)))./(r.^2+r_j.^-2-2.*r.*r_j.^-1.* ...
    cos(theta-theta_j(t)))));
u_theta_func = @(r,theta,t) -((-1./(4.*pi).*(2.*r-2.*r_j.* ...
    cos(theta-theta_j(t)))./(r.^2+r_j.^2-2.*r.*r_j.* ...
    cos(theta-theta_j(t))))-(-1./(4.*pi).*(2.*r-2.*r_j.^-1.* ...
    cos(theta-theta_j(t)))./(r.^2+r_j.^-2-2.*r.*r_j.^-1.* ...
    cos(theta-theta_j(t)))));

time = -1;   %resetting time for ftle

[Xg,Yg] = meshgrid(xcoor, ycoor);           %gas particles positions
[Xg_new,Yg_new] = meshgrid(xcoor, ycoor);   %updated positions
boundaryg = zeros(nx,ny);

for t = t_start:d*dt:t_end*d
        time = time+1;
        T_span = time*dt;

        Xg = Xg_new;
        Yg = Yg_new;

        [thetag, rg] = cart2pol(Xg,Yg);

        u_r = u_r_func(rg,thetag,t);
        u_theta = u_theta_func(rg,thetag,t);
        u = u_r.*cos(thetag)-u_theta.*sin(thetag);
        v = u_r.*sin(thetag)+u_theta.*cos(thetag);

        Ug = u*d; %velocity if the gas particles
        Vg = v*d;

        for i = 1:ny
            for j = 1:nx
```

```
122                 if domain(i,j) == 1 && boundaryg(i,j) == 0
123                     %calculating k1 for Runge-Kutta
124                     k1x(i,j) = dt*Ug(i,j);
125                     k1y(i,j) = dt*Vg(i,j);
126                 end
127             end
128         end
129
130         [thetag1, rg1] = cart2pol(Xg+k1x/2,Yg+k1y/2);
131         u_r1 = u_r_func(rg1,thetag1,t+0.5*dt*d);
132         u_theta1 = u_theta_func(rg1,thetag1,t+0.5*dt*d);
133         u1 = u_r1.*cos(thetag1)-u_theta1.*sin(thetag1);
134         v1 = u_r1.*sin(thetag1)+u_theta1.*cos(thetag1);
135
136         Ug1 = u1*d;
137         Vg1 = v1*d;
138
139         for i = 1:ny
140             for j = 1:nx
141                 if domain(i,j) == 1 && boundaryg(i,j) == 0
142                     %calculating k2 for Runge-Kutta
143                     k2x(i,j)=dt*Ug1(i,j);
144                     k2y(i,j)=dt*Vg1(i,j);
145                 end
146             end
147         end
148
149         [thetag2, rg2] = cart2pol(Xg+k2x/2,Yg+k2y/2);
150         u_r2 = u_r_func(rg2,thetag2,t+0.5*dt*d);
151         u_theta2 = u_theta_func(rg2,thetag2,t+0.5*dt*d);
152         u2 = u_r2.*cos(thetag2)-u_theta2.*sin(thetag2);
153         v2 = u_r2.*sin(thetag2)+u_theta2.*cos(thetag2);
154
155         Ug2 = u2*d;
156         Vg2 = v2*d;
157
158         for i = 1:ny
159             for j = 1:nx
160                 if domain(i,j) == 1 && boundaryg(i,j) == 0
161                     %calculating k3 for Runge-Kutta
162                     k3x(i,j)=dt*Ug2(i,j);
163                     k3y(i,j)=dt*Vg2(i,j);
164                 end
165             end
```

```
166         end
167
168         [thetag3, rg3] = cart2pol(Xg+k3x,Yg+k3y);
169         u_r3 = u_r_func(rg3,thetag3,t+dt*d);
170         u_theta3 = u_theta_func(rg3,thetag3,t+dt*d);
171         u3 = u_r3.*cos(thetag3)-u_theta3.*sin(thetag3);
172         v3 = u_r3.*sin(thetag3)+u_theta3.*cos(thetag3);
173
174         Ug3 = u3*d;
175         Vg3 = v3*d;
176
177         for i = 1:ny
178             for j = 1:nx
179                 if domain(i,j) == 1 && boundaryg(i,j) == 0
180                     %calculating k4 for Runge-Kutta
181                     k4x(i,j)=dt*Ug3(i,j);
182                     k4y(i,j)=dt*Vg3(i,j);
183                 end
184             end
185         end
186
187         for i = 1:ny
188             for j = 1:nx
189                 if domain(i,j) == 1 && boundaryg(i,j) == 0
190                     %new position
191                     Xg_new(i,j)=Xg(i,j)+1/6*(k1x(i,j)+ ...
192                         2*k2x(i,j)+2*k3x(i,j)+k4x(i,j));
193                     Yg_new(i,j)=Yg(i,j)+1/6*(k1y(i,j)+ ...
194                         2*k2y(i,j)+2*k3y(i,j)+k4y(i,j));
195                 end
196             end
197         end
198
199         [thetag1, rg1] = cart2pol(Xg_new,Yg_new);
200
201         for i = 1:ny
202             for j = 1:nx
203                 if rg1(i,j) ≥ 1 && domain(i,j) == 1 && ...
204                         boundaryg(i,j) == 0
205                     %setting the particles which left the domain ...
206                         %on the wall
207                     Xg_new(i,j) = ...
                           1*cos(thetag(i,j)+0.5*(thetag1(i,j)-thetag(i,j)));
```

```
208                     Yg_new(i,j) = ...
                            1*sin(thetag(i,j)+0.5*(thetag1(i,j)-thetag(i,j)));
209                     boundaryg(i,j) = 1;
210                 end
211             end
212         end
213
214         %calculating ftle
215         for i=1:ny
216             for j=1:nx
217                 if domain(i,j) == 1
218                     if (j-1)*(j-nx)<0 && (i-1)*(i-ny)<0
219                         A11=(Xg(i,j+1)-Xg(i,j-1))/(xcoor(j+1)-xcoor(j-1));
220                         A12=(Xg(i-1,j)-Xg(i+1,j))/(ycoor(i-1)-ycoor(i+1));
221                         A21=(Yg(i,j+1)-Yg(i,j-1))/(xcoor(j+1)-xcoor(j-1));
222                         A22=(Yg(i-1,j)-Yg(i+1,j))/(ycoor(i-1)-ycoor(i+1));
223                         A =[A11 A12;A21 A22];
224                         sigma(i,j)=log(norm(A))/(T_span);
225                     else
226                         sigma(i,j)=0;
227                     end
228                 end
229             end
230         end
231
232         Z = sigma;
233
234         if corotating == 1
235             %rotate the frame to watch in the corotating frame
236             Xgplot = rg.*cos(thetag-theta_j(t));
237             Ygplot = rg.*sin(thetag-theta_j(t));
238         else
239             Xgplot = Xg;
240             Ygplot = Yg;
241         end
242
243         if write == 1
244             save([num2str(name) num2str(t*d/dt) '.mat'], 'Xg', 'Yg', ...
                        'Ug', 'Vg', 'Z');
245         end
246
247         if write == 0
248             figure(1)
249             H=gcf;
```

```
250             set(H,'Position',[200 200 500 500]);
251             plot(Xgplot(1:5:nx,1:5:ny),Ygplot(1:5:nx,1:5:ny), 'k.')
252             text(0.7,0.9,['t = ' num2str(t*d) 's'], 'color', 'black');
253             rectangle('position',[-1,-1,2,2],'curvature', ...
                    [1,1],'LineWidth',2);
254             view(2)
255             shading interp
256             xlabel('x');
257             ylabel('y');
258             title('Gas particles');
259             axis([-1 1 -1 1])
260             daspect([1 1 1])
261             drawnow
262 %             hold on
263 %             for i = 1:5:nx
264 %                 for j =1:5:ny
265 %                     if domain(i,j) == 1
266 %                         plot(Xgplot(i,j),Ygplot(i,j), 'r.')
267 %                     end
268 %                 end
269 %             end
270
271             figure(2)
272             H=gcf;
273             set(H,'Position',[800 200 500 500]);
274             surf(X0,Y0,Z)
275             text(0.7,0.9,['t = ' num2str(t*d) 's'], 'color', 'black');
276             rectangle('position',[-1,-1,2,2],'curvature', ...
                    [1,1],'LineWidth',2);
277             %caxis([0 0.4])
278             view(2)
279             shading interp
280             xlabel('x');
281             ylabel('y');
282             title('FTLE field');
283             axis([-1 1 -1 1])
284             daspect([1 1 1])
285             drawnow
286         end
287
288 end
```

## A.2 INERTIAL PARTICLES

```
clear all; clc; close all

%--input--
nx = 400;               %number of gridpoints
ny = 400;

xstart = -1;            %domain
xlast  =  1;
ystart =  1;
ylast  = -1;

St = 0.5;               %Stokes number of the inertial particles

dt = 0.025;             %integration time step

write = 0;              %1 for writing datafiles, 0 for plots ...
    without writing

r_j = 0.5;              %radial position of the point vortex

T_start = 0;            %starting time
T_end = 100;            %end time of the run

d = 1;                  %direction of ftle, forward = 1, backward = -1

if d == 1
    direction = 'forward';
elseif d == -1
    direction = 'backward';
end

name = ['LCS_' direction '_0025_'];     %name of datafile

corotating = 0;         %setting frame, 1 = corotating, 0 = rotating


%----------

xcoor = linspace(xstart, xlast, nx);        %vectors for meshgrid
ycoor = linspace(ystart, ylast, ny);

[X0,Y0] = meshgrid(xcoor, ycoor);           %initial positions of ...
    particles
```

```
[Xp,Yp] = meshgrid(xcoor, ycoor);           %positions of inertial ...
   particles
[Xp_new,Yp_new] = meshgrid(xcoor, ycoor);

[theta, r] = cart2pol(Xp,Yp);               %polar coordinates

%defining matrices
Ugp = zeros(ny,nx);
Vgp = zeros(ny,nx);
Ugp1 = zeros(ny,nx);
Vgp1 = zeros(ny,nx);
Ugp2 = zeros(ny,nx);
Vgp2 = zeros(ny,nx);

Up_new = zeros(ny,nx);
Vp_new = zeros(ny,nx);
Up = zeros(ny,nx);
Vp = zeros(ny,nx);
Up2 = zeros(ny,nx);
Vp2 = zeros(ny,nx);
Up1 = zeros(ny,nx);
Vp1 = zeros(ny,nx);
Xp1 = zeros(ny,nx);
Yp1 = zeros(ny,nx);

Xg1 = zeros(ny,nx);
Yg1 = zeros(ny,nx);

sigma = zeros(ny,nx);

domain = zeros(ny,nx);
boundaryp = zeros(ny,nx);
boundaryg = zeros(ny,nx);

Z = zeros(ny,nx);

for i = 1:nx
    for j =1:ny
        if sqrt(X0(i,j)^2+Y0(i,j)^2) ≤ 1
            domain(i,j) = 1;                %the particles in the ...
               domain (circle with r=1) will be traced
        end
    end
end
```

```
84
85 theta_j = @(t) (1/(2*pi))*(1/(1-r_j^2))*t;      %angular position of ...
      point vortex
86 theta_jt = (1/(2*pi))*(1/(1-r_j^2));           %angular velocity of ...
      point vortex
87
88 u_r_func = @(r,theta,t) (1./r).*((-1./(4.*pi).*(2.*r.*r_j.* ...
89     sin(theta-theta_j(t))./(r.^2+r_j.^2-2.*r.*r_j.* ...
90     cos(theta-theta_j(t))))-(-1./(4.*pi).*(2.*r.*r_j.^-1.* ...
91     sin(theta-theta_j(t))./(r.^2+r_j.^-2-2.*r.*r_j.^-1.* ...
92     cos(theta-theta_j(t)))));
93 u_theta_func = @(r,theta,t) -((-1./(4.*pi).*(2.*r-2.*r_j.* ...
94     cos(theta-theta_j(t))./(r.^2+r_j.^2-2.*r.*r_j.* ...
95     cos(theta-theta_j(t))))-(-1./(4.*pi).*(2.*r-2.*r_j.^-1.* ...
96     cos(theta-theta_j(t))./(r.^2+r_j.^-2-2.*r.*r_j.^-1.* ...
97     cos(theta-theta_j(t)))));
98 time = 0;
99
100 for T = T_start:dt*d:d*T_end
101
102     time = time+1;   %resetting time for ftle
103     T_span = time*dt;
104
105     Xp = Xp_new;
106     Yp = Yp_new;
107     Up2 = Up1;
108     Vp2 = Vp1;
109     Up1 = Up;
110     Vp1 = Vp;
111     Up = Up_new;
112     Vp = Vp_new;
113     Ugp2 = Ugp1;
114     Ugp1 = Ugp;
115     Vgp2 = Vgp1;
116     Vgp1 = Vgp;
117
118     [thetap, rp] = cart2pol(Xp,Yp);                %polar coordinates
119
120     u_r = u_r_func(rp,thetap,T);
121     u_theta = u_theta_func(rp,thetap,T);
122     u = u_r.*cos(thetap)-u_theta.*sin(thetap);      %describing the ...
          velocities in x and y directions
123     v = u_r.*sin(thetap)+u_theta.*cos(thetap);
124
```

```
125     Ugp = d*u;  %velocity of the gas at position of the inertial ...
            particles
126     Vgp = d*v;
127
128         if T == 0
129             for i = 1:ny
130                 for j = 1:nx
131                     if domain(i,j) == 1
132                         %inertial particles Euler method
133                         Up(i,j) = Ugp(i,j);    %initial velocities
134                         Vp(i,j) = Vgp(i,j);
135                     end
136                 end
137             end
138
139             for i = 1:ny
140                 for j = 1:nx
141                     if domain(i,j) == 1 && boundaryp(i,j) == 0
142                         %inertial particles Euler method
143                         Xp_new(i,j) = Xp(i,j)+dt*Up(i,j);
144                         Yp_new(i,j) = Yp(i,j)+dt*Vp(i,j);
145
146                         Up_new(i,j) = Up(i,j) + ...
                                dt*(1/St)*(Ugp(i,j)-Up(i,j));
147                         Vp_new(i,j) = Vp(i,j) + ...
                                dt*(1/St)*(Vgp(i,j)-Vp(i,j));
148                     end
149                 end
150             end
151         elseif T == dt*d
152             for i = 1:ny
153                 for j = 1:nx
154                     if domain(i,j) == 1 && boundaryp(i,j) == 0
155                         %inertial particles two step adams bashforth
156                         Xp_new(i,j)= ...
                                Xp(i,j)+1.5*dt*Up(i,j)-0.5*dt*Up1(i,j);
157                         Yp_new(i,j)= ...
                                Yp(i,j)+1.5*dt*Vp(i,j)-0.5*dt*Vp1(i,j);
158
159                         Up_new(i,j)= ...
                                Up(i,j)+1.5*dt*(1/St)*(Ugp(i,j)-Up(i,j))- ...
160                              0.5*dt*(1/St)*(Ugp1(i,j)-Up1(i,j));
161                         Vp_new(i,j)= ...
                                Vp(i,j)+1.5*dt*(1/St)*(Vgp(i,j)-Vp(i,j))- ...
```

```
                        0.5*dt*(1/St)*(Vgp1(i,j)-Vp1(i,j));
                end
            end
        end
    else
        for i = 1:ny
            for j = 1:nx
                if domain(i,j) == 1 && boundaryp(i,j) == 0
                    %heavy particles three step adams bashforth
                    Xp_new(i,j)=Xp(i,j)+dt*((23/12)*Up(i,j)- ...
                        (4/3)*Up1(i,j)+(5/12)*Up2(i,j));
                    Yp_new(i,j)=Yp(i,j)+dt*((23/12)*Vp(i,j)- ...
                        (4/3)*Vp1(i,j)+(5/12)*Vp2(i,j));

                    Up_new(i,j)=Up(i,j)+dt*((23/12)*((1/St)* ...
                        (Ugp(i,j)-Up(i,j)))-(4/3)*((1/St)* ...
                        (Ugp1(i,j)-Up1(i,j)))+(5/12)*((1/St)* ...
                        (Ugp2(i,j)-Up2(i,j))));
                    Vp_new(i,j)=Vp(i,j)+dt*((23/12)*((1/St)* ...
                        (Vgp(i,j)-Vp(i,j)))-(4/3)*((1/St)* ...
                        (Vgp1(i,j)-Vp1(i,j)))+(5/12)*((1/St)* ...
                        (Vgp2(i,j)-Vp2(i,j))));
                end
            end
        end
    end

    [thetap1, rp1] = cart2pol(Xp_new,Yp_new);

    for i = 1:ny
        for j = 1:nx
            if rp1(i,j) ≥ 1 && domain(i,j) == 1 && ...
               boundaryp(i,j) == 0
                %setting the particles which left the domain on ...
                   the wall
                Xp_new(i,j) = ...
                   1*cos(thetap(i,j)+0.5*(thetap1(i,j)-thetap(i,j)));
                Yp_new(i,j) = ...
                   1*sin(thetap(i,j)+0.5*(thetap1(i,j)-thetap(i,j)));
                boundaryp(i,j) = 1;
            end
            if boundaryp(i,j) == 1
                %velocities for particles on the wall are zero
                Up_new(i,j) = 0;
```

```
                    Vp_new(i,j) = 0;
                    Up(i,j) = 0;
                    Vp(i,j) = 0;
                    Up1(i,j) = 0;
                    Vp1(i,j) = 0;
                    Up2(i,j) = 0;
                    Vp2(i,j) = 0;
                end
            end
        end

if corotating == 1
    %rotate the frame to watch in the corotating frame
    Xplot = rp.*cos(thetap-theta_j(T));
    Yplot = rp.*sin(thetap-theta_j(T));
else
    Xplot = Xp;
    Yplot = Yp;
end

        %calculating ftle
        for i=1:ny
            for j=1:nx
                if domain(i,j) == 1
                    if (j-1)*(j-nx)<0 && (i-1)*(i-ny)<0
                        A11=(Xp(i,j+1)-Xp(i,j-1))/(xcoor(j+1)-xcoor(j-1));
                        A12=(Xp(i-1,j)-Xp(i+1,j))/(ycoor(i-1)-ycoor(i+1));
                        A21=(Yp(i,j+1)-Yp(i,j-1))/(xcoor(j+1)-xcoor(j-1));
                        A22=(Yp(i-1,j)-Yp(i+1,j))/(ycoor(i-1)-ycoor(i+1));
                        A =[A11 A12;A21 A22];
                        sigma(i,j)=log(norm(A))/(T_span);
                    else
                        sigma(i,j)=0;
                    end
                end
            end
        end

        Z = sigma;

    if write == 0
        figure(1)
        clf(1);
        H=gcf;
```

```
246         set(H,'Position',[200 200 500 500]);
247         %         hold on
248         %only plot the particles in the domain (slower)
249 %         for i = 1:5:nx
250 %             for j =1:5:ny
251 %                 if domain(i,j) == 1
252 %                     plot(Xgplot(i,j),Ygplot(i,j), 'r.')
253 %                 end
254 %             end
255 %         end
256         %plot the whole grid
257         plot(Xplot(1:5:nx,1:5:ny),Yplot(1:5:nx,1:5:ny), 'k.')
258         text(0.7,0.9,['t = ' num2str(T*d) 's'], 'color', 'black');
259         rectangle('position',[-1,-1,2,2],'curvature', ...
              [1,1],'LineWidth',2);
260         caxis([0 0.2])
261         view(2)
262         shading interp
263         xlabel('x');
264         ylabel('x');
265         title('Inertial particles');
266         axis([-1 1 -1 1])
267         daspect([1 1 1])
268         drawnow
269
270
271         figure(2)
272         H=gcf;
273         set(H,'Position',[800 200 500 500]);
274         daspect([1 1 1])
275         surf(X0,Y0,Z)
276         text(0.7,0.9,['t = ' num2str(T*d) 's'], 'color', 'black');
277         rectangle('position',[-1,-1,2,2],'curvature', ...
              [1,1],'LineWidth',2);
278         caxis([0 0.2])
279         view(2)
280         shading interp
281         xlabel('x');
282         ylabel('y');
283         title('FTLE field');
284         axis([-1 1 -1 1])
285         daspect([1 1 1])
286         drawnow
287 %         for i = 1:5:200
```

```
288 %             for j = 1:5:200
289 %                 if domain(i,j) == 1
290 %                 plot(Xg(i,j),Yg(i,j), 'w.')
291 %                 end
292 %             end
293 %         end
294
295     end
296
297     if write == 1
298         save([num2str(name) num2str(St*10) '_' num2str((d*T)/dt)], ...
                'Xp', 'Yp', 'Up', 'Vp', 'Z');
299     end
300 end
```

# APPENDIX

## INERTIAL PARTICLE VELOCITIES

# INERTIAL PARTICLE VELOCITIES

## B.1 VELOCITY IN X-DIRECTION

```
Up_func = @(x,y,t) -((y*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - ...
    r_j^2)*pi) - atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - ...
    (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) ...
    + (2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2)) ...
    + (x*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) + ...
    (r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2) - ...
    St*(((x*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) + (2*sqrt(x^2 ...
    + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2) + ...
          (y*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +         ...
    (r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2))*     ...
    ((y^2*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
    (2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^(3/2) ...
    -       (-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +         ...
    (2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
```

```
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))))/sqrt(x^2 + y^2) - ...
      (2*x*y*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^2 - ...
      (y*(-(((2*y)/sqrt(x^2 + y^2) - (2*x*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*(x^2 + y^2)))/(4*pi*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j))) +          ((2*y)/sqrt(x^2 + y^2) - ...
(2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/(x^2 + ...
y^2))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) +          ...
((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)*(2*y - (2*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) - (2*x*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/             (r_j*sqrt(x^2 + ...
y^2))))/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)^2) -          ...
((2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))*(2*y - (2*r_j*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/sqrt(x^2 + y^2) - (2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) ...
- atan2(y,x)))/             sqrt(x^2 + y^2)))/(4*pi*(r_j^2 + x^2 ...
+ y^2 - 2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))^2)))/sqrt(x^2 + y^2) +       (x*((x*cos(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)) -          (r_j*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) - ...
         (y*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + x^2 + y^2 - ...
(2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)) ...
+          (r_j*y*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) + ...
         (sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x))*(2*y - (2*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) - (2*x*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*sqrt(x^2 + y^2))))/          ...
(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 ...
- r_j^2)*pi) - atan2(y,x)))/r_j)^2) - (r_j*sqrt(x^2 + ...
y^2)*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x))*           (2*y - ...
```

```
(2*r_j*y*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + y^2) ...
- (2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + ...
y^2)))/          (2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))^2)))/(x^2 + y^2)) + ...
    (-((y*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2)) ...
+       (x*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2))*     ...
((x*y*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^(3/2) ...
-       (2*x^2*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^2 + ...
      (-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2) - ...
      (y*(-(((2*x)/sqrt(x^2 + y^2) + (2*y*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*(x^2 + y^2)))/(4*pi*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j))) +          ((2*x)/sqrt(x^2 + y^2) + ...
(2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/(x^2 + ...
y^2))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) +          ...
((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)*(2*x - (2*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) + (2*y*sin(t/(2*(1 - ...
```

```
    r_j^2)*pi) - atan2(y,x)))/             (r_j*sqrt(x^2 + ...
    y^2))))/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)^2) -          ...
    ((2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))*(2*x - (2*r_j*x*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/sqrt(x^2 + y^2) + (2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) ...
    - atan2(y,x)))/             sqrt(x^2 + y^2)))/(4*pi*(r_j^2 + x^2 ...
    + y^2 - 2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))^2)))/sqrt(x^2 + y^2) +       (x*(-((y*cos(t/(2*(1 - ...
    r_j^2)*pi) - atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + ...
    x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/r_j))) +          (r_j*y*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
    2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) - ...
             (x*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + x^2 + y^2 - ...
    (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)) ...
    +          (r_j*x*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
    2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) + ...
             (sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x))*(2*x - (2*x*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) + (2*y*sin(t/(2*(1 - ...
    r_j^2)*pi) - atan2(y,x)))/(r_j*sqrt(x^2 + y^2))))/          ...
    (2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 ...
    - r_j^2)*pi) - atan2(y,x)))/r_j)^2) - (r_j*sqrt(x^2 + ...
    y^2)*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x))*           (2*x - ...
    (2*r_j*x*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + y^2) ...
    + (2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + ...
    y^2)))/          (2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))^2)))/(x^2 + y^2)));
```

## B.2 VELOCITY IN Y-DIRECTION

```
1 Vp_func = @(x,y,t) (x*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - ...
    r_j^2)*pi) - atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - ...
    (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) ...
    +  (2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
    y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2) + ...
      (y*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
    atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
```

```
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +      ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2) -   ...
St*(((x*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2) + ...
      (y*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2))*     ...
(-((x*y*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + ...
y^2)^(3/2)) -       (2*y^2*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - ...
(2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) ...
+          (r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^2 + ...
      (-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2) + ...
      (x*(-(((2*y)/sqrt(x^2 + y^2) - (2*x*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*(x^2 + y^2)))/(4*pi*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j))) +          ((2*y)/sqrt(x^2 + y^2) - ...
(2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/(x^2 + ...
y^2))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) +          ...
((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)*(2*y - (2*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) - (2*x*sin(t/(2*(1 - ...
```

```
r_j^2)*pi) - atan2(y,x)))/             (r_j*sqrt(x^2 + ...
y^2))))/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)^2) -          ...
((2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))*(2*y - (2*r_j*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/sqrt(x^2 + y^2) - (2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) ...
- atan2(y,x)))/             sqrt(x^2 + y^2)))/(4*pi*(r_j^2 + x^2 ...
+ y^2 - 2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))^2)))/sqrt(x^2 + y^2) +       (y*((x*cos(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)) -          (r_j*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) - ...
         (y*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + x^2 + y^2 - ...
(2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)) ...
+          (r_j*y*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) + ...
         (sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x))*(2*y - (2*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) - (2*x*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*sqrt(x^2 + y^2))))/          ...
(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 ...
- r_j^2)*pi) - atan2(y,x)))/r_j)^2) - (r_j*sqrt(x^2 + ...
y^2)*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x))*           (2*y - ...
(2*r_j*y*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + y^2) ...
- (2*r_j*x*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + ...
y^2)))/          (2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))^2)))/(x^2 + y^2)) + ...
    (-((y*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/sqrt(x^2 + y^2)) ...
+       (x*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +          ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2))*     ...
(-((x^2*(-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
```

```
atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + ...
y^2)^(3/2)) +       (-((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/r_j)/(4*pi*(r_j^(-2) + x^2 + y^2 - ...
(2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) ...
+         (2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))))/sqrt(x^2 + y^2) - ...
      (2*x*y*(-((sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j))) +           ...
(r_j*sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x))))))/(x^2 + y^2)^2 + ...
      (x*(-(((2*x)/sqrt(x^2 + y^2) + (2*y*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*(x^2 + y^2)))/(4*pi*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j))) +          ((2*x)/sqrt(x^2 + y^2) + ...
(2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/(x^2 + ...
y^2))/(4*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) +          ...
((2*sqrt(x^2 + y^2) - (2*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j)*(2*x - (2*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) + (2*y*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/             (r_j*sqrt(x^2 + ...
y^2))))/(4*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)^2) -          ...
((2*sqrt(x^2 + y^2) - 2*r_j*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))*(2*x - (2*r_j*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/sqrt(x^2 + y^2) + (2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) ...
- atan2(y,x)))/             sqrt(x^2 + y^2)))/(4*pi*(r_j^2 + x^2 ...
+ y^2 - 2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))^2)))/sqrt(x^2 + y^2) +       (y*(-((y*cos(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + ...
x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/r_j))) +          (r_j*y*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) - ...
         (x*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*r_j*pi*sqrt(x^2 + y^2)*(r_j^(-2) + x^2 + y^2 - ...
(2*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/r_j)) ...
```

```
+          (r_j*x*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(2*pi*sqrt(x^2 + y^2)*(r_j^2 + x^2 + y^2 - ...
2*r_j*sqrt(x^2 + y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))) + ...
         (sqrt(x^2 + y^2)*sin(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x))*(2*x - (2*x*cos(t/(2*(1 - r_j^2)*pi) - ...
atan2(y,x)))/(r_j*sqrt(x^2 + y^2)) + (2*y*sin(t/(2*(1 - ...
r_j^2)*pi) - atan2(y,x)))/(r_j*sqrt(x^2 + y^2))))/          ...
(2*r_j*pi*(r_j^(-2) + x^2 + y^2 - (2*sqrt(x^2 + y^2)*cos(t/(2*(1 ...
- r_j^2)*pi) - atan2(y,x)))/r_j)^2) - (r_j*sqrt(x^2 + ...
y^2)*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x))*           (2*x - ...
(2*r_j*x*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + y^2) ...
+ (2*r_j*y*sin(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))/sqrt(x^2 + ...
y^2)))/          (2*pi*(r_j^2 + x^2 + y^2 - 2*r_j*sqrt(x^2 + ...
y^2)*cos(t/(2*(1 - r_j^2)*pi) - atan2(y,x)))^2)))/(x^2 + y^2)));
```