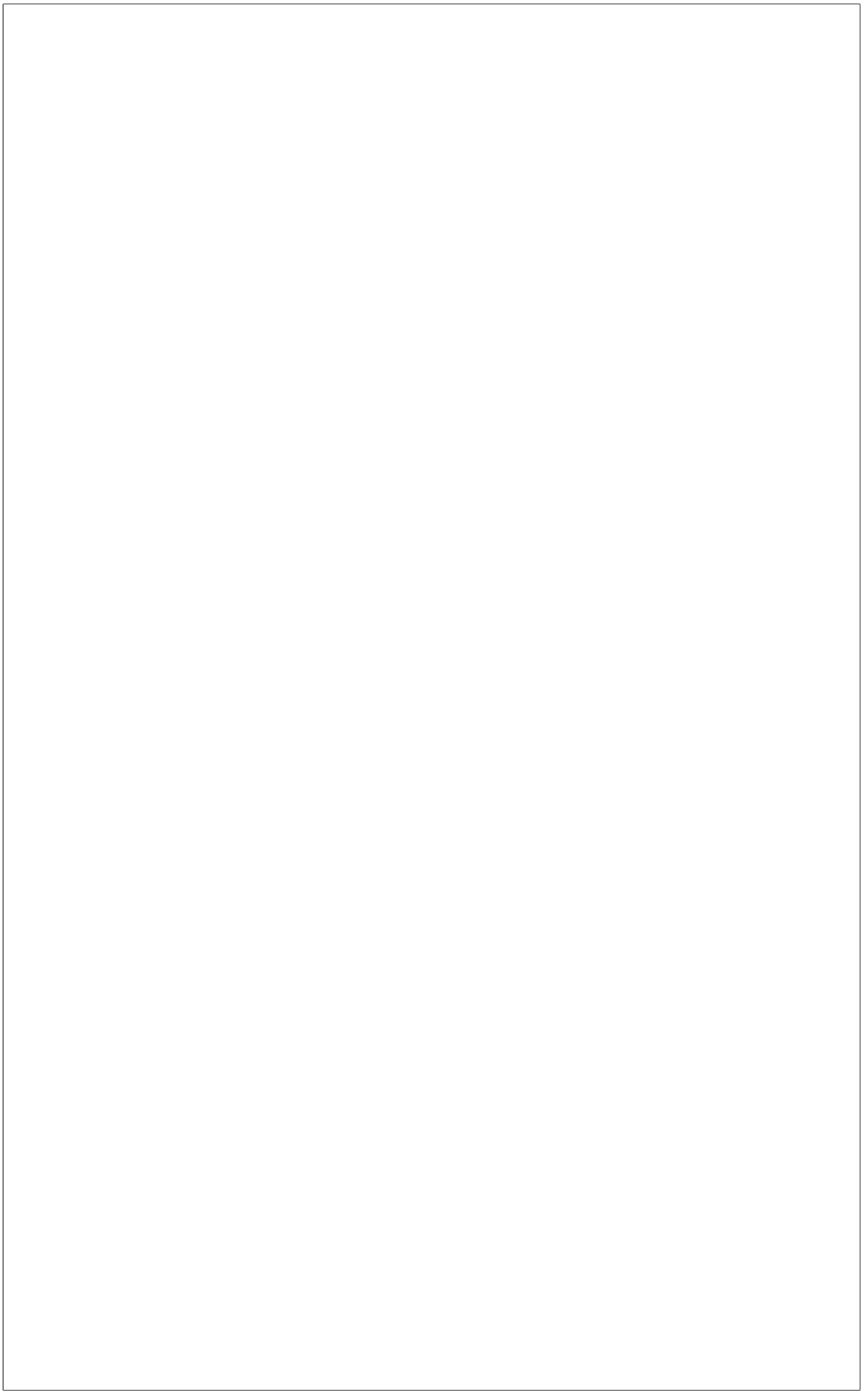
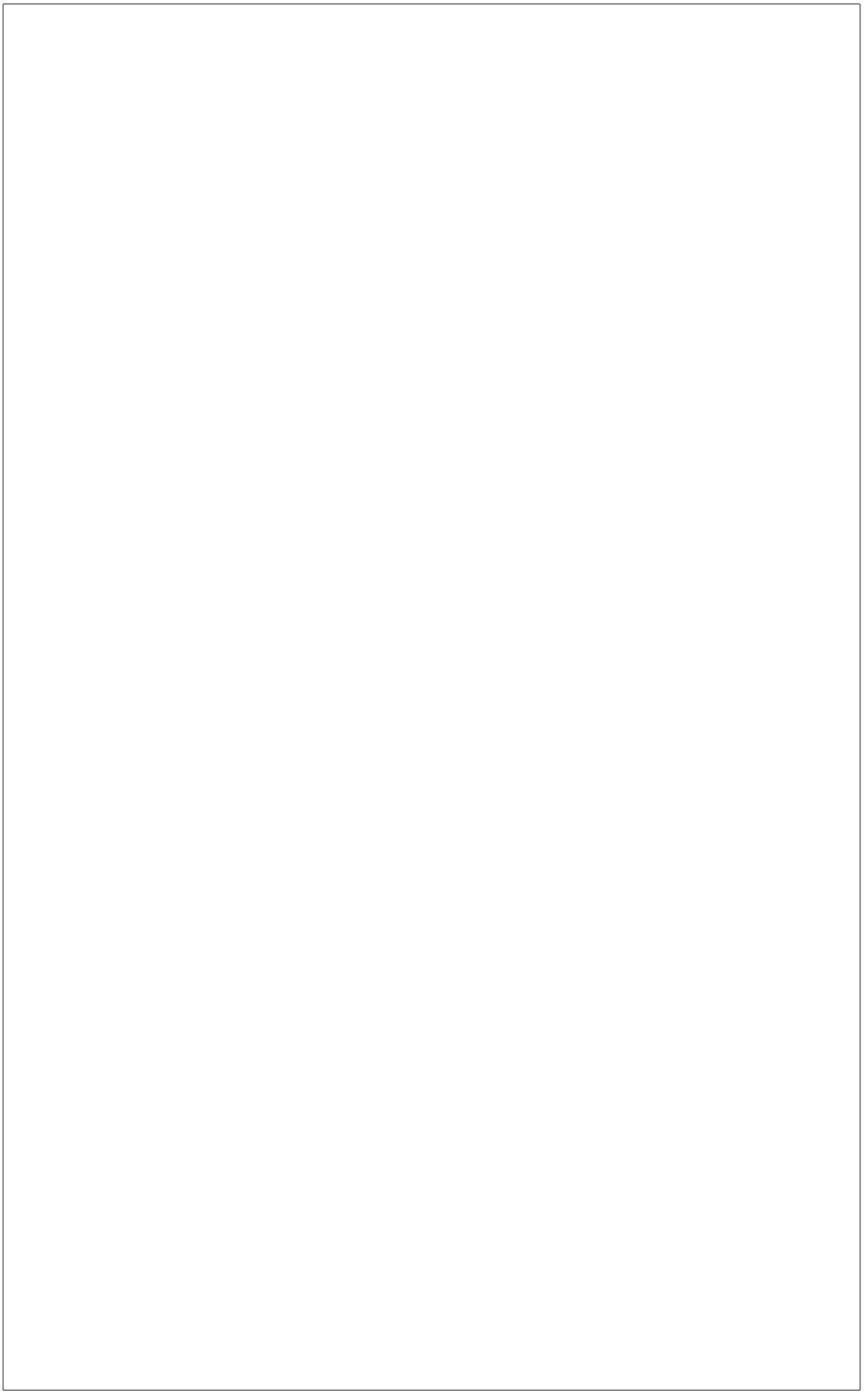
Abstract



Sammendrag



Preface

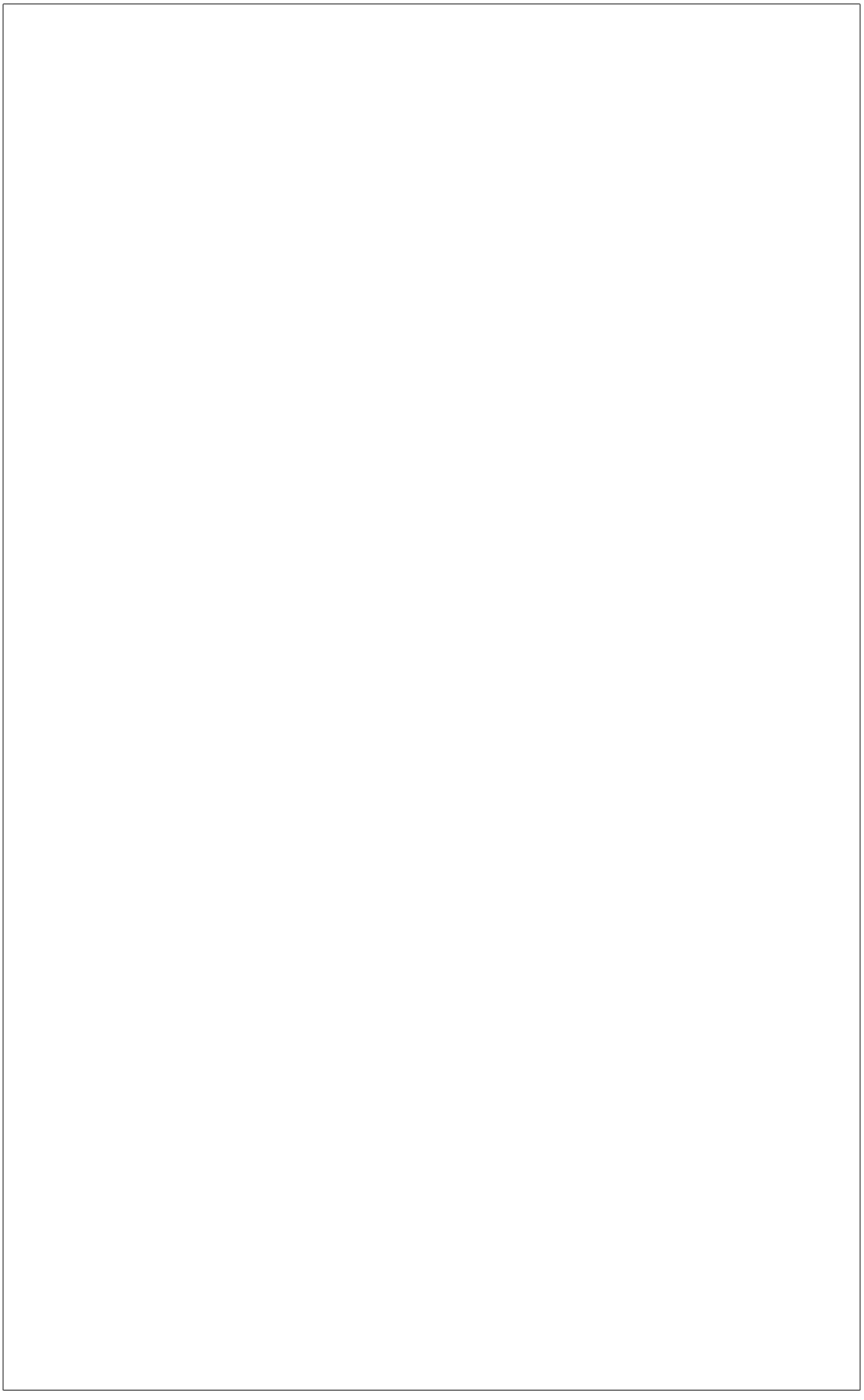
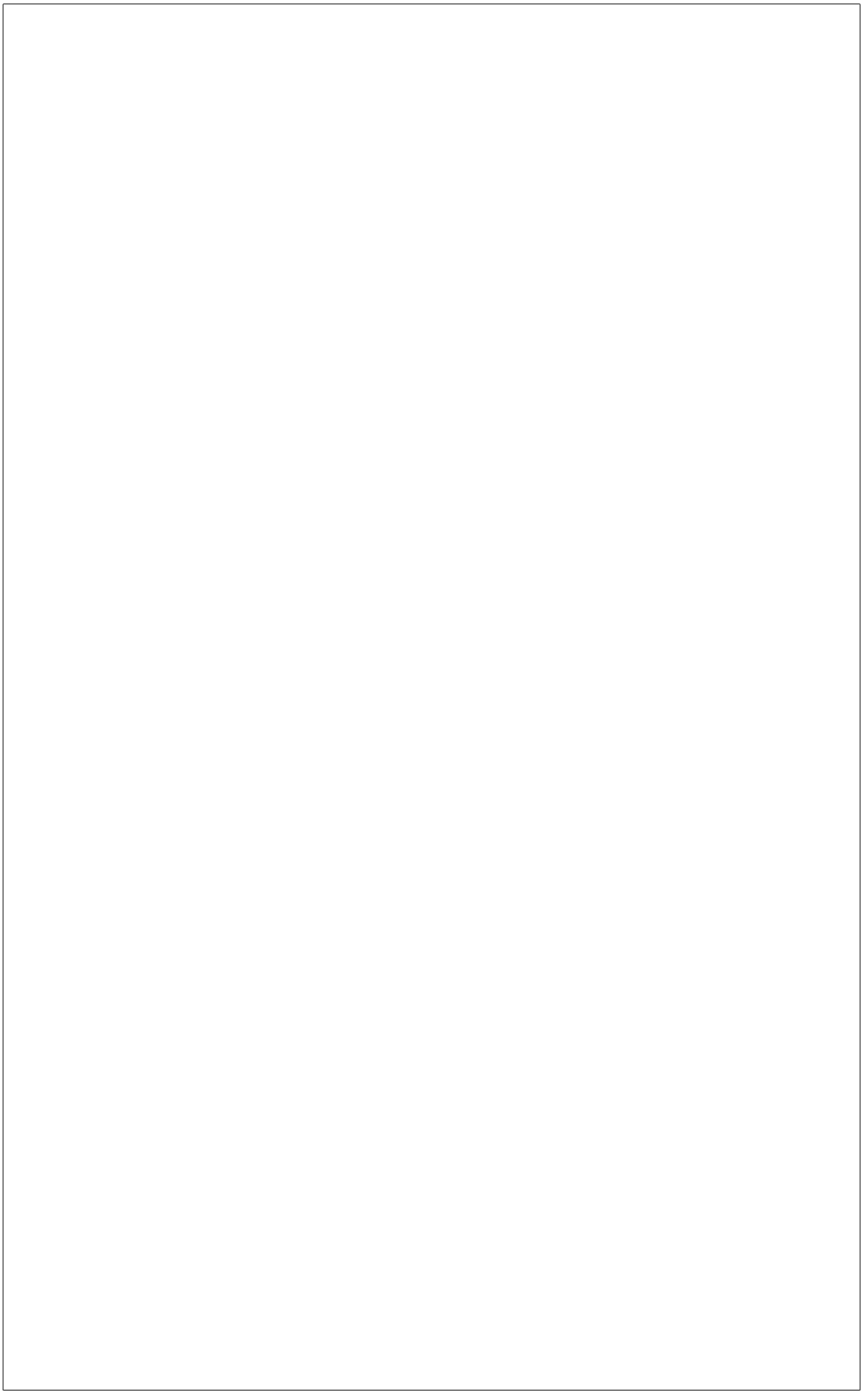
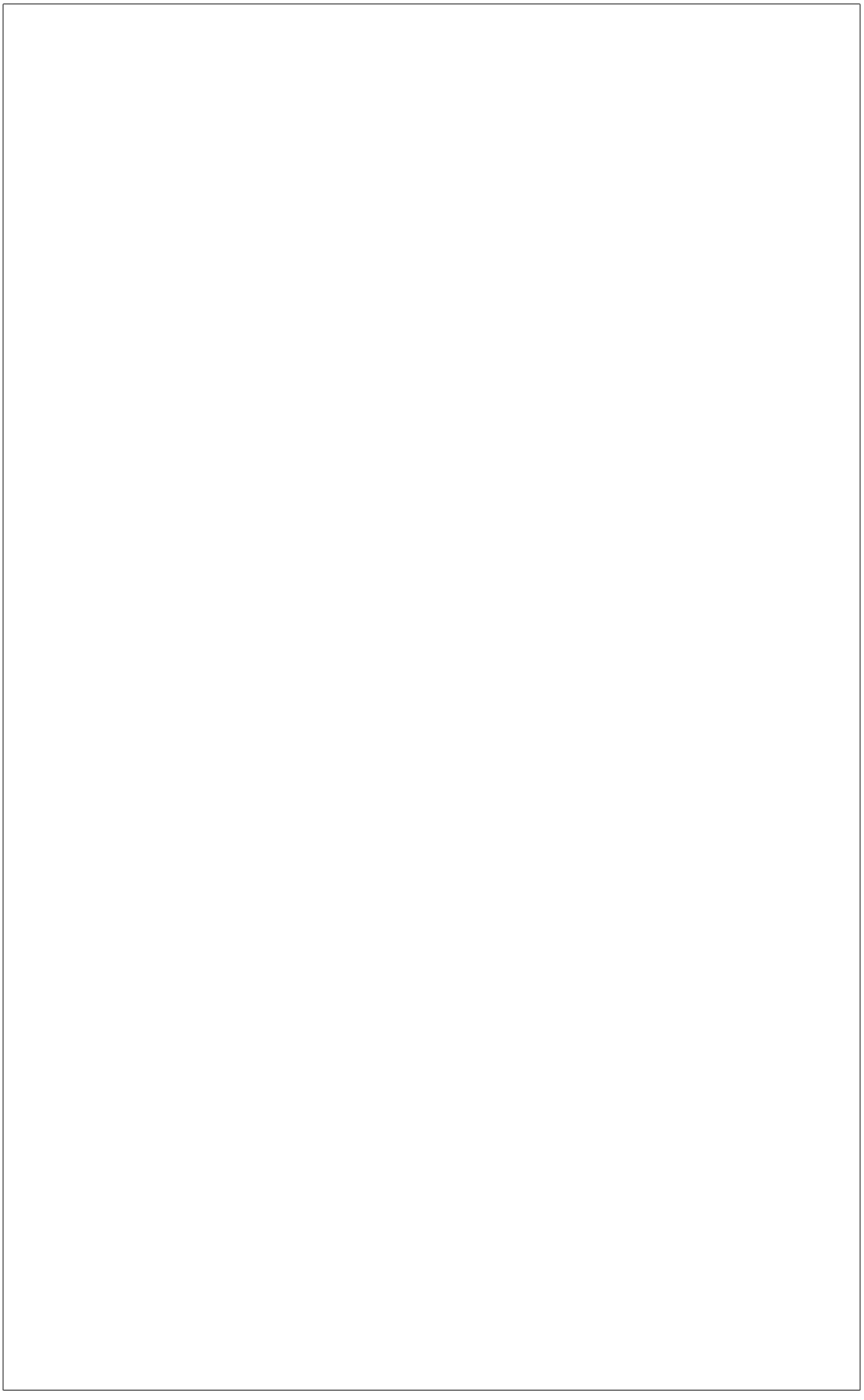


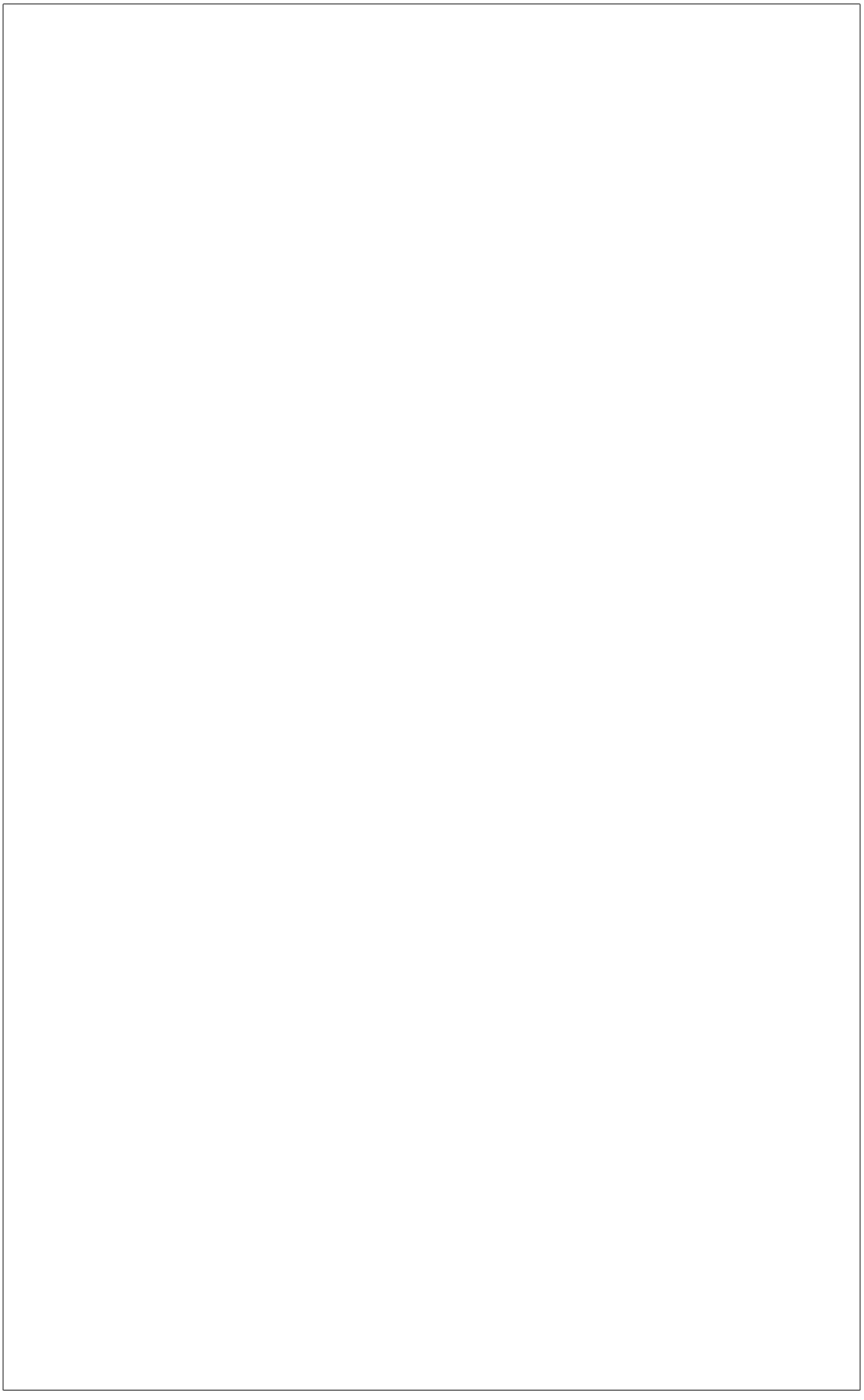
Table of Contents



List of Figures



List of Tables



Notation

Newton's notation is used for differentiation with respect to time, i.e.:

$$\dot{f}(t) \equiv \frac{\mathrm{d}f(t)}{\mathrm{d}t}.$$

Vectors are denoted by lowercase, upright, bold letters, like this:

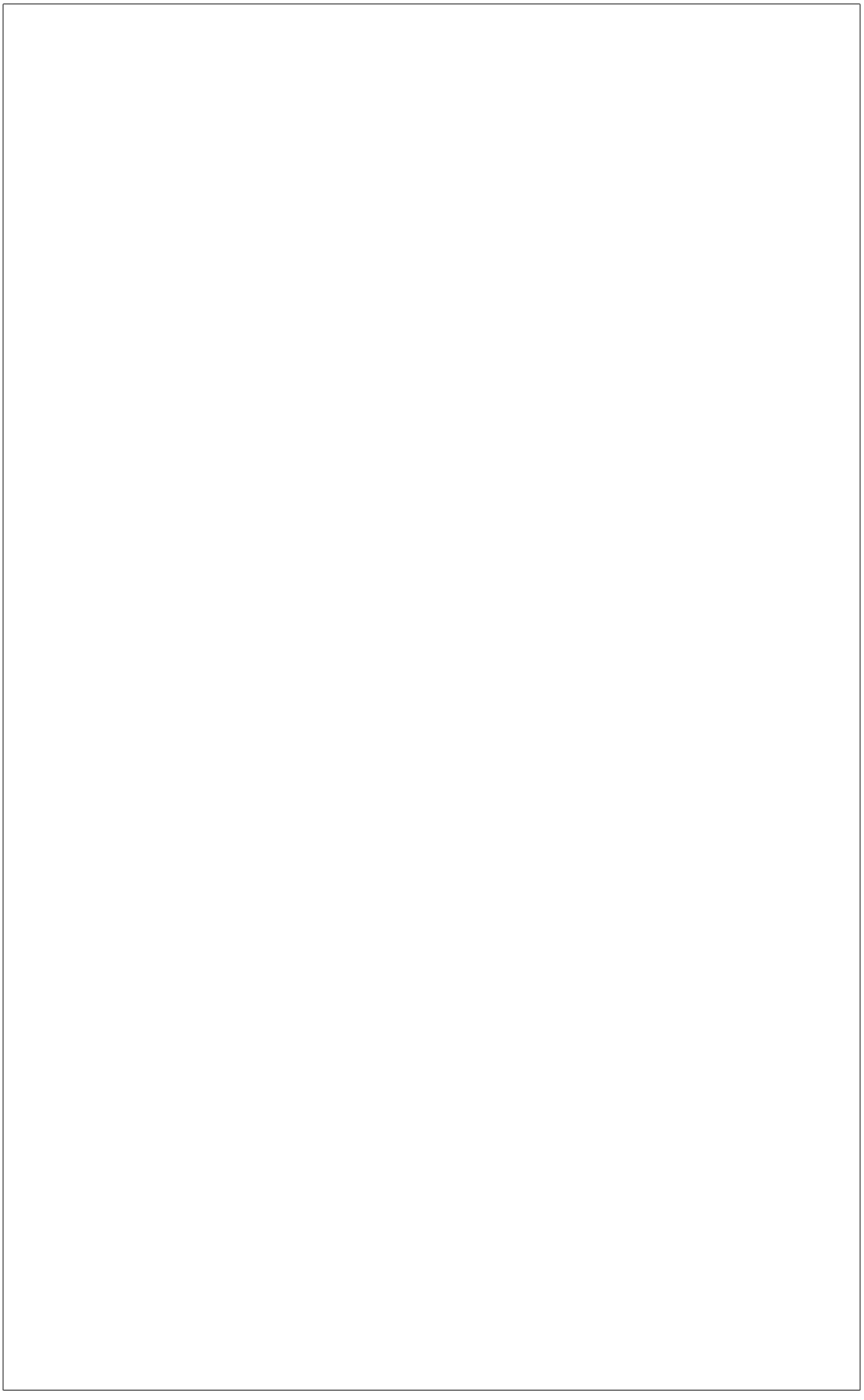
$$\boldsymbol{\xi} = \left(\xi_1, \xi_2, \ldots, \xi_n\right).$$

The Euclidean norm of a vector $\boldsymbol{\xi} \in \mathbb{R}^n$ is denoted by:

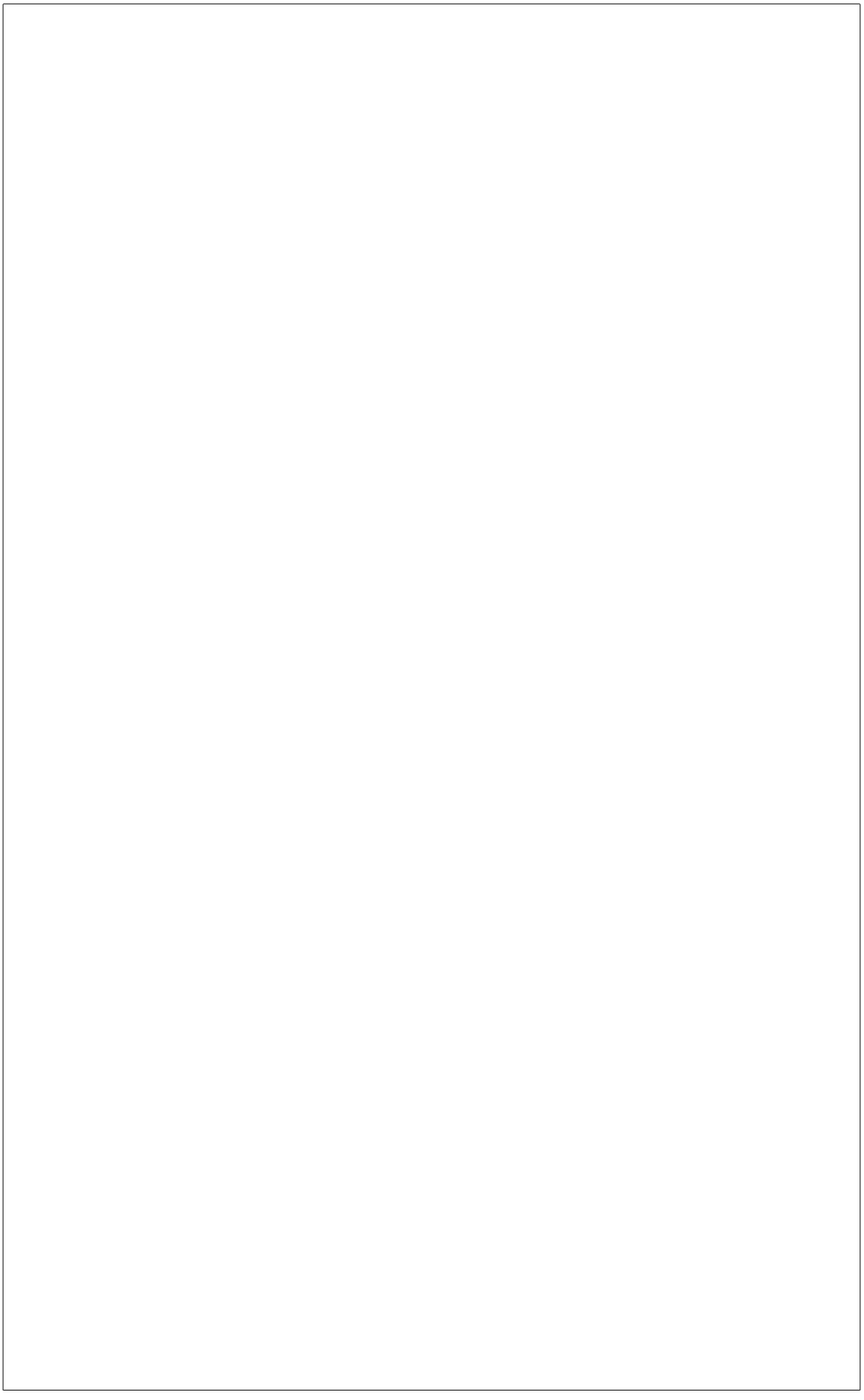
$$\|\boldsymbol{\xi}\| = \sqrt{\xi_1^2 + \xi_2^2 + \ldots + \xi_n^2}.$$

Matrices and matrix representations of rank-2 tensors are denoted by uppercase, upright, bold letters, as follows:

$$\mathbf{A} = (a_{i,j}) = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{pmatrix}.$$



1 Introduction



2 Theory

2.1 Solving systems of ordinary differential equations

In physics, like other sciences, modeling a system often equates to solving an initial value problem. An initial value problem can be described in terms of an ordinary differential equation (hereafter abbreviated to ODE) of the form

$$\dot{x}(t) = f\big(t, x(t)\big), \quad x(t_0) = x_0, \tag{2.1}$$

where x is an unknown function (scalar or vector) of time t. The function f is defined on an open subset Ω of $\mathbb{R} \times \mathbb{R}^n$, where n is the number of spatial dimensions; that is, the number of components of x. The initial condition (t_0, x_0) is a point in the domain of f, i.e., $(t_0, x_0) \in \Omega$. In higher dimensions (namely, $n > 1$), the differential equation (2.1) generally extends to a coupled family of ODEs

$$\dot{x}_i(t) = f_i\big(t, x_1(t), x_2(t), \ldots, x_n(t)\big), \quad x_i(t_0) = x_{i,0}, \quad i = 1, \ldots, n. \tag{2.2}$$

The system is nonlinear if the function f in equation (2.1), or, if at least one of the functions $\{f_i\}$ in equation (2.2), is nonlinear in one or more of its arguments. For the sake of notational simplicity, the discussion to follow in the rest of this section is based on the one-dimensional case, that is, system (2.1), for $n = 1$. However, all of the considerations also hold for $n > 1$.

Say that the solution of system (2.1) is sought at some time t_f. In order to approximate said solution numerically, the time variable must be discretized first. This is frequently done by defining

$$t_j = t_0 + j \cdot h, \tag{2.3}$$

where t_j is the time level j for integer j, and h is some increment which is smaller than $t_f - t_0$. Typically, the time increment is chosen such that an integer number of step lengts h equals the difference $t_f - t_0$. With the discretized time, the numerical solution of system (2.1) is found by successive applications of some numerical integration method. The Runge-Kutta family of numerical methods for ODE systems is a common choice, and will be elaborated upon in greater detail in section 2.1.1.

All numerical integration schemes fall into one of two categories; explicit and implicit methods. Explicit methods are characterized by computing the state of the system at a later time, based on the state of the system at the current time (in some cases, the state at earlier times are also considered). Implicit methods, however, involve the solution of an equation in which both the current and the later state of the system are involved. Thus, a generic, explicit method for computing the state of the system at time $t + h$, given its state at t, can be expressed as

$$x(t + h) = F\big(x(t)\big), \tag{2.4a}$$

while, for implicit methods, an equation of the sort

$$G\big(x(t), x(t + h)\big) = 0, \tag{2.4b}$$

is solved to find $x(t+h)$.

In general, implicit methods require the solution of a linear system at every time step. Clearly, implicit methods are more computationally demanding than explicit methods. The main selling point of implicit methods is that they are more numerically stable than explicit methods. This property means that implicit methods are particularly well-suited for *stiff* systems, i.e., physical systems with highly disparate time scales (Hairer and Wanner 1996, p.2). For such systems, most explicit methods are unstable, unless the time step h is made exceptionally small, rendering these methods practically useless. For *nonstiff* systems, however, implicit methods behave similarly to their explicit analogues in terms of numerical accuracy and convergence properties.

Irrespective of which numerical integration method is employed, one obtains an approximation of the true solution of the system (2.1) *at* the discrete time levels, that is,

$$x_j \approx x(t_j), \tag{2.5}$$

where $x(t)$ is the exact solution at time t. The accuracy of the approximation, however, depends on both the numerical integration method and the time step length h used for the temporal discretization. One way of obtaining approximations of the true solution *inbetween* the discrete time levels is by means of interpolation — a numerical technique which will be elaborated upon in section 2.1.2. For nonlinear systems, analytical solutions usually do not exist. Thus, such systems are often analyzed by means of numerical methods.

2.1.1 *The Runge-Kutta family of numerical ODE solvers*

In numerical analysis, the Runge-Kutta family of methods is a popular collection of implicit and explicit iterative methods, used in temporal discretization in order to obtain numerical approximations of the *true* solutions of systems like (2.1). The German mathematicians C. Runge and M.W. Kutta developed the first of the family's methods at the turn of the twentieth century (Hairer, Nørsett, and Wanner 1993, p.134). The general outline of what is now known as a Runge-Kutta method is as follows:

Definition 1 *(Runge-Kutta methods).*
Let s be an integer and $\{a_{i,j}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$ be real coefficients.
Let h be the numerical step length used in the temporal discretization.
Then, the method

$$\begin{aligned} k_i &= f\left(t_n + c_i h, x_n + h\sum_{j=1}^{s} a_{i,j}k_j\right), \quad i = 1,\dots,s, \\ x_{n+1} &= x_n + h\sum_{i=1}^{s} b_i k_i, \end{aligned} \tag{2.6}$$

is called an *s-stage Runge-Kutta method* for the system (2.1).

The main reason to include multiple stages in a Runge-Kutta method is to improve the numerical accuracy of the computed solutions. The *order* of a Runge-Kutta method can be defined as follows:

Definition 2 *(Order of Runge-Kutta methods).*
A Runge-Kutta method, given by equation (2.6), is of *order p* if, for sufficiently smooth systems (2.1), the local error e_n scales as h^{p+1}. That is:

$$e_n = \|x_n - u_{n-1}(t_n)\| \leq K\, h^{p+1} \tag{2.7}$$

where $u_{n-1}(t)$ is the exact solution of the ODE in system (2.1) at time t, subject to the initial condition $u_{n-1}(t_{n-1}) = x_{n-1}$, and K is a numerical constant. This is true, if the Taylor series for the exact solution $u_{n-1}(t_n)$ and the numerical solution x_n coincide up to (and including) the term h^p.

The *global* error

$$E_n = x_n - x(t_n), \tag{2.8}$$

where $x(t)$ is the exact solution of system (2.1) at time t, accumulated by n repeated applications of the numerical method, can be estimated by

$$|E_n| \leq C \sum_{l=1}^{n} |e_l|, \tag{2.9}$$

where C is a numerical constant, depending on both the right hand side of the ODE in system (2.1) and the difference $t_n - t_0$. Making use of definition 2, the global error is limited from above by

$$\begin{aligned} |E_n| &\leq C \sum_{l=1}^{n} |e_l| \leq C \sum_{l=1}^{n} |K_l|\, h^{p+1} \leq C \max_l \left\{|K_l|\right\} n\, h^{p+1} \\ &\leq C \max_l \left\{|K_l|\right\} \frac{t_n - t_0}{h}\, h^{p+1} \leq \widetilde{K}\, h^p, \end{aligned} \tag{2.10}$$

where $\widetilde{K}$ is a numerical constant. Equation (2.10) demonstrates that, for a p-th order Runge-Kutta method, the global error can be expected to scale as h^p.

In definition 1, the matrix $(a_{i,j})$ is commonly called the *Runge-Kutta matrix*, while the coefficients $\{b_i\}$ and $\{c_i\}$ are known as the *weights* and *nodes*, respectively. Since the 1960s, it has been customary to represent Runge-Kutta methods, given by equation (2.6), symbolically, by means of mnemonic devices known as Butcher tableaus (Hairer, Nørsett, and Wanner 1993, p.134). The Butcher tableau for a general s-stage Runge-Kutta method, as introduced in definition 1, is presented in table 2.1. For explicit Runge-Kutta methods, the Runge-Kutta matrix $(a_{i,j})$ is lower triangular. Similarly, for fully implicit Runge-Kutta methods, the Runge-Kutta matrix is upper triangular. The difference between explicit and implicit methods is outlined in equation (2.4).

During the first half of the twentieth century, a substantial amount of research was conducted in order to develop numerically robust, high-order, explicit Runge-Kutta methods. The idea

Table 2.1: Butcher tableau representing a generic s-stage Runge-Kutta method.

c_1	$a_{1,1}$	$a_{1,2}$	$\dots$	$a_{1,s}$
c_2	$a_{2,1}$	$a_{2,2}$	$\dots$	$a_{2,s}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s}$
	b_1	b_2	$\dots$	b_s

was that using such methods would mean one could resort to larger time increments h without sacrificing precision in the computed solution. However, the required number of stages s grows quicker than linearly as a function of the required order p. It has been proven that, for $p \geq 5$, no explicit Runge-Kutta method of order p with $s = p$ stages exists (Hairer, Nørsett, and Wanner 1993, p.173). This is one of the reasons for the attention shift from the latter half of the 1950s and onwards, towards so-called *embedded* Runge-Kutta methods.

The basic idea of embedded Runge-Kutta methods is that they, aside from the numerical approximation x_{n+1}, yield a second approximation $\widehat{x}_{n+1}$. The difference between the two approximations then provides an estimate of the local error of the less precise result, which can be used for automatic step size control (Hairer, Nørsett, and Wanner 1993, pp.167–168). The trick is to construct two independent, explicit Runge-Kutta methods which both use the *same* function evaluations. This results in practically obtaining the two solutions for the price of one, in terms of computational complexity. The Butcher tableau of a generic, embedded, explicit Runge-Kutta method is illustrated in table 2.2.

For embedded methods, the coefficients are tuned such that

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \tag{2.11a}$$

is of order p, and

$$\widehat{x}_{n+1} = x_n + h \sum_{i=1}^{s} \widehat{b}_i k_i \tag{2.11b}$$

is of order $\widehat{p}$, typically with $\widehat{p} = p+1$. Which of the solutions is used to continue the numerical integration, depends on the integration method in question. In the following, the solution which is *not* used to continue the integration, will be referred to as the *interpolant* solution.

There exists an abundance of Runge-Kutta methods; many of which are fine-tuned for specific constraints, such as problems of varying degrees of stiffness. Based on prior investigations — such as the work done by Løken (2017) — using explicit, high order, embedded Runge-Kutta methods to compute Lagrangian coherent structures (which will be elaborated upon in ***SETT INN REF***) consistently yields accurate solutions at lower computational cost than the most common fixed stepsize methods. Accordingly, the Dormand-Prince 8(7) method — consisting

Table 2.2: Butcher tableau representation a generic, embedded, explicit Runge-Kutta method.

0					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_s	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s-1}$	
	b_1	b_2	$\dots$	b_{s-1}	b_s
	$\widehat{b}_1$	$\widehat{b}_2$	$\dots$	$\widehat{b}_{s-1}$	$\widehat{b}_s$

of an eighth order solution with a seventh order interpolant — was chosen as the single, multipurpose, numerical ODE solver for this project.

Note that the concept of *order* is less well-defined for embedded methods than for fixed stepsize methods, as a direct consequence of the adaptive time step. Although the *local* errors of each integration step scale as per equation (2.7), the bound on the *global* (i.e., observable) error suggested in equation (2.10) is invalid, as the time step is, in principle, different for each integration step. A Butcher tableau representation of the Dormand-Prince 8(7) method is available in table 2.3, which has been typeset in landscape mode for the reader's convenience. Details on how the dynamic time step was implemented will be presented in ***SETT INN REF***.

2.1.2 *Spline interpolation of discrete data*

As all naturally ocurring physical systems can only be known partially, either by means of partial measurements or grid based model output, interpolating (i.e., estimating) the measurement data becomes a requirement when describing the dynamics of systems which depend on measurement data from inbetween the sampling or grid points. Spline interpolation involves approximating a discretely sampled function by a series of piecewise defined polynomials. According to Stoer and Bulirsch (2002, p.93), spline interpolation is a popular tool within the field of numerical analysis, due to yielding smooth interpolation curves with limited interpolation error when using low degree polynomial pieces. Furthermore, the local nature of spline interpolated functions means that such functions are less prone to oscillatory behaviour when using high order polynomials. This is in stark constrast to regular, global polynomial interpolation, which exhibits strong global dependence on local properties. In particular, if the function to be approximated is badly behaved anywhere within the interval of approximation, then the approximation by polynomial interpolation is poor everywhere (de Boor 1978, p.17).

Table 2.3: Butcher tableau for the Dormand-Prince 8(7) embedded Runge-Kutta method. The b coefficients give an 8^{th}-order accurate solution used to continue the integration. The $\widehat{b}$ coefficients yield a 7^{th}-order interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The provided coefficients are rational approximations, accurate to about 24 significant decimal digits. For reference, see Prince and Dormand (1981).

0													
$\frac{1}{18}$	$\frac{1}{18}$												
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$											
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$										
$\frac{5}{16}$	$\frac{5}{16}$	0	$\frac{-75}{64}$	$\frac{75}{64}$									
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$								
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736358}{692538347}$	$\frac{-28693883}{1125000000}$	$\frac{23124283}{1800000000}$							
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$\frac{-180193667}{1043307555}$						
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$\frac{-433636366}{683701615}$	$\frac{-421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$					
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$\frac{-37695042795}{15268766246}$	$\frac{-309121744}{1061227803}$	$\frac{-12992083}{490766935}$	$\frac{6005943493}{2108947689}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$				
$\frac{1201146811}{1299019798}$	$\frac{-1028468189}{846180014}$	0	0	$\frac{8478235783}{508512851}$	$\frac{1311729495}{1432422823}$	$\frac{-10304129995}{1701304382}$	$\frac{-48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{-45442868181}{339846796}$	$\frac{3065993473}{5917172653}$			
1	$\frac{185892177}{718116043}$	0	0	$\frac{-3185094517}{667107341}$	$\frac{-477755414}{1098053517}$	$\frac{-703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{-4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$		
1	$\frac{403863854}{491063109}$	0	0	$\frac{-5068492393}{434740067}$	$\frac{-411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{-13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{-160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	
	$\frac{1400451}{335480064}$	0	0	0	0	$\frac{-59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{56129285}{797845732}$	$\frac{-1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$\frac{-528747749}{2220607170}$	$\frac{1}{4}$
	$\frac{13451932}{455176623}$	0	0	0	0	$\frac{-808719486}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$\frac{-3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	0

A generic interpolation problem can be described in terms of a family of functions

$$\Theta(\mathbf{x}; \beta_0, \ldots, \beta_n), \tag{2.12}$$

each of which characterized by the $n+1$ parameters $\{\beta_i\}$, with $\mathbf{x}$ containing the independent variables of the problem. Given a set of $n+1$ discrete measurements — each defined by a set of coordinates and function values $(\mathbf{x}_i, f_i)$ where $\mathbf{x}_i \neq \mathbf{x}_j$ for $i \neq j$ and $f_i = f(\mathbf{x}_i)$ — the interpolation problem reduces to finding parameters $\{\beta_i\}$ such that

$$\Theta(\mathbf{x}; \beta_0, \ldots, \beta_n) = f_i, \quad i = 0, \ldots, n \tag{2.13}$$

is satisfied. According to Stoer and Bulirsch (2002, pp.38–39), spline interpolation problems (amongst others) can be classified as linear interpolation problems, meaning that the family of interpolation functions (cf. equation (2.12)) can be expressed as

$$\Theta(\mathbf{x}; \beta_0, \ldots, \beta_n) = \sum_{i=0}^{n} \beta_i \Theta_i(\mathbf{x}). \tag{2.14}$$

In the following, let the coordinates $\{\mathbf{x}_i\}$, function values $\{f_i\}$ and sampling points $\{(\mathbf{x}_i, f_i)\}$ be denoted by *support abscissas*, *support ordinates* and *support points*, respectively.

Solving an interpolation problem by means of spline interpolation is done by determining the set of parameters $\{\beta_i\}$ of equations (2.13) and (2.14), with the family of functions $\{\Theta_i\}$ limited to spline functions. These functions, often denoted as *splines*, are connected through the use of a partition. Consider the one-dimensional case for the sake of notational simplicity — the considerations to follow also hold for higher dimensions, but invariably introduces notational clutter. The partition

$$\Delta: \quad \{a = x_0 < x_1 < \cdots < x_n = b\} \tag{2.15}$$

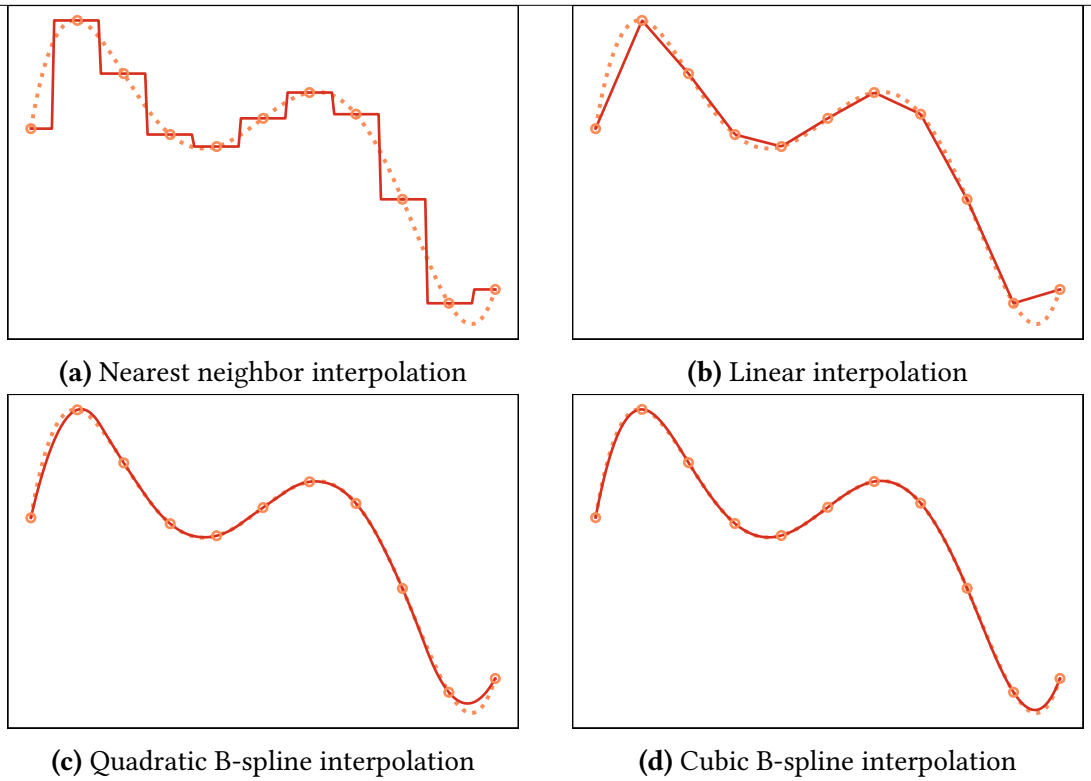
of the closed interval $[a, b]$ determines the domains of the piecewise polynomial spline functions $\mathcal{S}$ in the set $\mathcal{S}_\Delta$. These spline functions are joined at support abscissas, which, in the context of splines, are called *knots*.

Stoer and Bulirsch (2002, p.107) define a *piecewise polynomial function* as follows:

Definition 3 *(Piecewise polynomial functions).*
Let f be a real-valued function which, for each $i = 0, \ldots, n-1$, when restricted to the subinterval (x_i, x_{i+1}) of the partition given in equation (2.15) corresponds to a polynomial $p_i(x)$ of degree less than or equal to $r-1$.

In order to obtain a one-to-one correspondence between the function f and the polynomial sequence $\big(p_0(x), p_1(x), \ldots, p_{n-1}(x)\big)$, *define* f at the knots $\{x_i\}_{i=0}^{n-1}$, so that the function becomes continuous from the right.

Accordingly, spline functions $\mathcal{S}_\Delta$ of degree k are polynomial functions of degree k which are $(k-1)$ times continuously differentiable at the interior knots, that is, $\{x_i\}_{i=1}^{n-1}$, of the partition Δ. These k^{th}-order polynomials are uniquely determined by $k+1$ coefficients, of which k are given by the $(k-1)$ derivatives and function values at their left bordering knot.



(a) Nearest neighbor interpolation

(b) Linear interpolation

(c) Quadratic B-spline interpolation

(d) Cubic B-spline interpolation

Figure 2.1: A selection of commonly used interpolation methods applied to a discretely sampled, high order polynomial (dashed). The sampling points (knots) are shown as hollow circles. Observe how higher order splines yield increasingly accurate and smooth interpolations.

B-splines are a family of nonnegative spline functions which have minimal support for any given degree, smoothness and domain partition. Furthermore, any spline function of a given degree can be expressed as a linear combination of B-splines of the same degree (Stoer and Bulirsch 2002, pp.107–110). Accordingly, B-splines provide the foundation of efficient and numerically stable computations of splines. A selection of commonly used interpolation methods applied to a discretely sampled, high order polynomial is shown in figure 2.1. From the figure, the increased precision of higher order splines when applied to continuous functions is readily apparent. Note, however, that higher order interpolation methods require more sampling points than lower order methods. In particular, at least $k + 1$ samples are required in order to construct a k^{th} order spline. In higher dimensions, that is, with data which depends on several other (independent) variables, the required amount of samples increases rapidly with the interpolation order. Subsequently, the use of cubic B-splines constitutes a popular method for general-purpose interpolation, providing a good balance between numerical accuracy and computational complexity.

2.2 The type of flow systems considered

We consider flow in three-dimensional dynamical systems of the form

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}), \quad \mathbf{x} \in \mathcal{U}, \quad t \in \mathcal{I}, \tag{2.16}$$

i.e., systems defined for the finite time interval $\mathcal{I}$ on an open, bounded subset $\mathcal{U}$ of $\mathbb{R}^3$. In addition, the velocity field $\mathbf{v}$ is assumed to be smooth in its arguments. Depending on the exact nature of the velocity field $\mathbf{v}$, analytical particle trajectories, that is, analytical solutions of system (2.16), may or may not exist. The flow particles are assumed to be infinitesimal and massless, i.e., non-interacting *tracers* of the overall circulation.

Consider a finite time interval $[t_0, t_1] \subset \mathcal{I}$ such that all tracer trajectories $\mathbf{x}(t; t_0, \mathbf{x}_0)$ in the system given by equation (2.16) are defined for all times $t \in [t_0, t_1]$. Then, the flow map is defined as

$$\boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0) = \mathbf{x}(t; t_0, \mathbf{x}_0), \tag{2.17}$$

that is, the flow map describes the movement of tracers from one point in time to another mathematically. In general, the flow map is as smooth as the underlying velocity field (cf. system (2.16)) (Farazmand and Haller 2012a). In Lagrangian flow analysis, the *Jacobian matrix* of the flow map $\boldsymbol{\phi}_{t_0}^{t}$ plays a significant role. Component-wise, the Jacobian matrix of a general vector-valued function $\mathbf{f}$ is defined as

$$(\boldsymbol{\nabla}\mathbf{f})_{i,j} = \frac{\partial f_i}{\partial x_j}, \quad \mathbf{f} = \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots\big), \tag{2.18}$$

which, for our three-dimensional flow, reduces to

$$\boldsymbol{\nabla}\mathbf{f} = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} \\ \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial z}. \end{pmatrix}. \tag{2.19}$$

Making use of the definition of the flow map (cf. equation (2.17)) in conjunction with equation (2.16), one finds the following ordinary differential equation which describes the time evolution of the flow map:

$$\dot{\boldsymbol{\phi}} = \mathbf{v}(t, \boldsymbol{\phi}), \tag{2.20}$$

where t_0, t and $\mathbf{x}_0$ have been omitted in order to avoid notational clutter. These are, however, implicit by context. As the nabla operator is time-independent, equation (2.20) immediately yields an ordinary differential equation for the time development of the directional derivative of the flow map, namely

$$\frac{\mathrm{d}}{\mathrm{d}t}\big(\hat{\mathbf{u}} \cdot \boldsymbol{\nabla}\big)\boldsymbol{\phi} = \big(\hat{\mathbf{u}} \cdot \boldsymbol{\nabla}\big)\mathbf{v}(t, \boldsymbol{\phi}), \tag{2.21}$$

which holds along any constant unit vector $\hat{\mathbf{u}}$. On a regular Cartesian grid, equation (2.21) provides a coupled set of ordinary differential equations describing the time evolution of each component of the Jacobian of the flow map:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \phi_i}{\partial x_j}\right) &= \sum_k \left.\frac{\partial v_i}{\partial x_k}\right|_{(t,\boldsymbol{\phi})} \left.\frac{\partial \phi_k}{\partial x_j}\right|_{t}, \\ \left.\frac{\partial \phi_i}{\partial x_j}\right|_{t_0} &= \delta_{ij}, \quad \mathbf{x}_0 \in \mathcal{U}, \quad t \in [t_0, t_1], \end{aligned} \tag{2.22}$$

where the Kronecker delta is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{2.23}$$

The initial conditions for the Jacobi components reflect the fact that, for a regular Cartesian grid, the directional derivative of the x coordinate in the x direction is 1, but zero in the y and z directions.

For sufficiently smooth velocity fields, the flow map Jacobian $\nabla \boldsymbol{\phi}_{t_0}^{t}$ can be computed, which allows for the right Cauchy-Green strain tensor field to be defined as

$$\mathbf{C}_{t_0}^{t}(\mathbf{x}_0) = \left(\nabla \boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0)\right)^{*} \left(\nabla \boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0)\right), \tag{2.24}$$

where the asterisk refers to the adjoint operation, which, because the Jacobian $\nabla \boldsymbol{\phi}_{t_0}^{t}$ is real-valued, equates to matrix transposition. Moreover, as the Jacobian of the flow map is invertible, the Cauchy-Green strain tensor $\mathbf{C}_{t_0}^{t}(\mathbf{x}_0)$ is symmetric and positive definite (Farazmand and Haller 2012a). Thus, it has three real, positive eigenvalues and orthogonal, real eigenvectors. Its eigenvalues λ_i and corresponding unit eigenvectors $\boldsymbol{\xi}_i$ are defined by

$$\begin{aligned} \mathbf{C}_{t_0}^{t}(\mathbf{x}_0)\boldsymbol{\xi}_i(\mathbf{x}_0) &= \lambda_i \boldsymbol{\xi}_i(\mathbf{x}_0), \quad i = 1, 2, 3, \\ \left\langle \boldsymbol{\xi}_i(\mathbf{x}_0), \boldsymbol{\xi}_j(\mathbf{x}_0) \right\rangle &= \delta_{ij}, \quad 0 < \lambda_1(\mathbf{x}_0) \leq \lambda_2(\mathbf{x}_0) \leq \lambda_3(\mathbf{x}_0), \end{aligned} \tag{2.25}$$

where the Kronecker delta is defined in equation (2.23), and the dependence of λ_i and $\boldsymbol{\xi}_i$ on t_0 and t has been suppressed, for the sake of notational transparency. The geometric interpretation of equation (2.25) is that a fluid element undergoes the most stretching along the $\boldsymbol{\xi}_3$ axis, less stretching along the $\boldsymbol{\xi}_2$ axis, and the least stretching along the $\boldsymbol{\xi}_1$ axis. This concept is shown in figure 2.2.

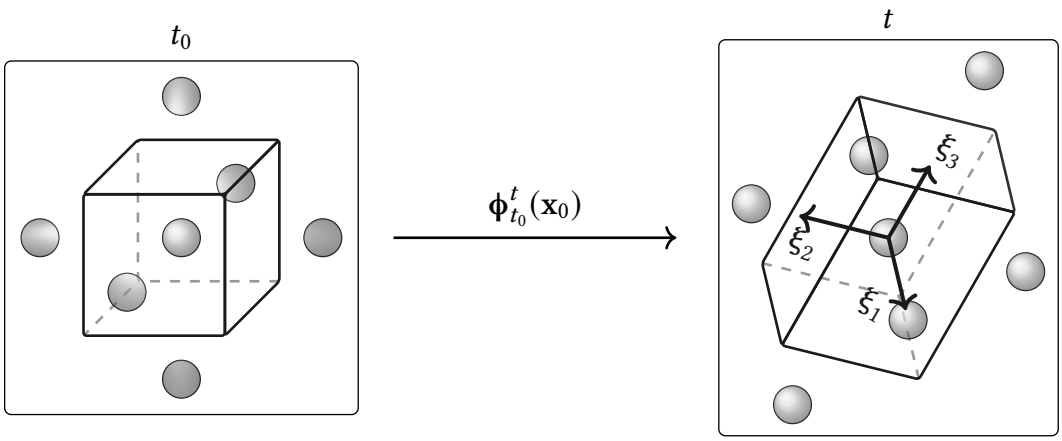


Figure 2.2: Geometrix interpretation of the eigenvectors of the Cauchy-Green strain tensor. The central unit cell is stretched and deformed under the flow map $\boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0)$. The local stretching is the largest in the direction of $\boldsymbol{\xi}_3$, the eigenvector which corresponds to the largest eigenvalue, λ_3, of the Cauchy-Green strain tensor, defined in equation (2.25). Along the $\boldsymbol{\xi}_i$ axes, the stretch factors are given by $\sqrt{\lambda_i}$, respectively.

As the stretch factors along the ξ_i axes are given by the square roots of the corresponding eigenvalues, for incompressible flow, the eigenvalues satisfy

$$\lambda_1(\mathbf{x}_0)\lambda_2(\mathbf{x}_0)\lambda_3(\mathbf{x}_0) = 1 \quad \forall \ \mathbf{x}_0 \in \mathcal{U}, \tag{2.26}$$

where, in the context of tracer advection, incompressibility is equivalent to the velocity field $\mathbf{v}$ being divergence-free (i.e., $\boldsymbol{\nabla} \cdot \mathbf{v} \equiv 0$ in system (2.16)).

2.3 Definition of Lagrangian coherent structures for three-dimensional flows

A necessary prerequisite for three-dimensional Lagrangian flow analysis is the concept of *material surfaces*, which Oettinger and Haller (2016) define as

Definition 4 *(Material surfaces).*
Consider a set of initial positions forming a two-dimensional surface $\mathcal{M}(t_0)$ at time t_0 in $\mathcal{U}$. Its time-t image, $\mathcal{M}(t)$, is obtained under the flow map as

$$\mathcal{M}(t) = \boldsymbol{\phi}_{t_0}^{t}\big(\mathcal{M}(t_0)\big). \tag{2.27}$$

The union of *all* time-t images, $\cup_{t\in[t_0,t_1]}\mathcal{M}(t)$, is a hypersurface in the extended phase space $\mathcal{U} \times \mathcal{I}$, called a *material surface*.

In the following, the entire material surface will be referred to by the notation $\mathcal{M}(t)$. Although no material surfaces can be crossed by tracers, only special material surfaces create coherence in the phase space $\mathcal{U}$ and thus act as observable transport barriers. Such material surfaces are generally referred to as *Lagrangian coherent structures* (henceforth abbreviated to LCSs).

Subsequently, LCSs can be described as time-evolving surfaces which shape coherent trajectory patterns in dynamical systems, defined over a finite time interval (Haller 2010). There are three main types of LCSs, namely *elliptic*, *hyperbolic* and *parabolic*. Rougly speaking, parabolic LCSs outline cores of jet-like trajectories, elliptic LCSs describe vortex boundaries, whereas hyperbolic LCSs are comprised of overall attractive or repelling manifolds. As such, hyperbolic LCSs practically act as organizing centers of observable tracer patterns (Onu, Huhn, and Haller 2015). Because hyperbolic LCSs provide the most readily applicable insight in terms of forecasting flow in e.g. oceanic currents, such structures have been the focus of this project.

2.3.1 *Hyperbolic LCSs in three dimensions*

The identification of LCSs for reliable forecasting requires sufficiency and necessity conditions, supported by mathematical theorems. Haller (2010) derived a variational LCS theory based on the Cauchy-Green strain tensor, defined by equation (2.24), from which the aforementioned conditions follow. The immediately relevant parts of Haller's theory are given in definitions 5–8 (Haller 2010).

Definition 5 *(Normally repellent material surfaces).*
A *normally repellent material surface* over the time interval $[t_0, t_1]$ is a compact material surface segment $\mathcal{M}(t)$ which is overall repelling, and on which the normal repulsion rate is greater than the tangential repulsion rate.

The required *compactness* of the material surface segment signifies that, in some sense, it must be topologically well-behaved. That the material surface is *overall repelling* means that nearby trajectories are repelled from, rather than attracted towards, the material surface. Lastly, requiring that the *normal* repulsion rate is greater than the *tangential* repulsion rate means that nearby trajectories are in fact driven away from the material surface, rather than being stretched along with it due to shear stress.

Definition 6 *(Repelling LCS).*
A *repelling LCS* over the time interval $[t_0, t_1]$ is a normally repelling material surface $\mathcal{M}(t_0)$ whose normal repulsion admits a pointwise non-degenerate maximum relative to any nearby material surface $\widehat{\mathcal{M}}(t_0)$.

Definition 7 *(Attracting LCS).*
An *attracting LCS* over the time interval $[t_0, t_1]$ is defined as a repelling LCS over the *backward* time interval $[t_1, t_0]$.

Definition 8 *(Hyperbolic LCS).*
A *hyperbolic LCS* over the time interval $[t_0, t_1]$ is a *repelling* or *attracting* LCS over the same time interval.

Note that the above definitions associate LCSs with the time interval $\mathcal{I}$ over which the dynamical system under consideration is known, or, at the very least, where information regarding the behaviour of tracers, is sought. Generally, LCSs obtained over a time interval $\mathcal{I}$ do not necessarily exist over different time intervals (Farazmand and Haller 2012a).

For sufficiently smooth three-dimensional flow, the above definitions can be summarized as a set of mathematical existence criteria, based on the Cauchy-Green strain tensor (cf. equation (2.24)) (Haller 2010; Farazmand and Haller 2012a; Karrasch 2012; Farazmand and Haller 2012b). These are given in theorem 1.

Theorem 1 *(Sufficient and necessary conditions for LCSs in three-dimensional flows).*
Consider a compact material surface $\mathcal{M}(t) \subset \mathcal{U}$ evolving over the time interval $[t_0, t_1]$. Then $\mathcal{M}(t)$ is a repelling LCS over $[t_0, t_1]$ if and only if all of the following holds for all initial conditions $\mathbf{x}_0 \in \mathcal{M}(t_0)$:

$$\lambda_2(\mathbf{x}_0) \neq \lambda_3(\mathbf{x}_0) > 1, \tag{2.28a}$$

$$\left\langle \xi_3(\mathbf{x}_0), \mathbf{H}_{\lambda_3}(\mathbf{x}_0)\xi_3(\mathbf{x}_0) \right\rangle < 0 \tag{2.28b}$$

$$\xi_3(\mathbf{x}_0) \perp \mathcal{M}(t_0), \tag{2.28c}$$

$$\left\langle \nabla\lambda_3(\mathbf{x}_0), \xi_3(\mathbf{x}_0) \right\rangle = 0. \tag{2.28d}$$

In theorem 1, $\langle \cdot, \cdot \rangle$ signifies the Euclidean inner product, and $\mathbf{H}_{\lambda_3}$ denotes the Hessian matrix of the largest eigenvalues of the Cauchy-Green strain tensor field. Component-wise, the Hessian matrix of a general, smooth, scalar-valued function f is defined as

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \tag{2.29}$$

which, for our three-dimensional flow, reduces to

$$\mathbf{H}_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{pmatrix}. \tag{2.30}$$

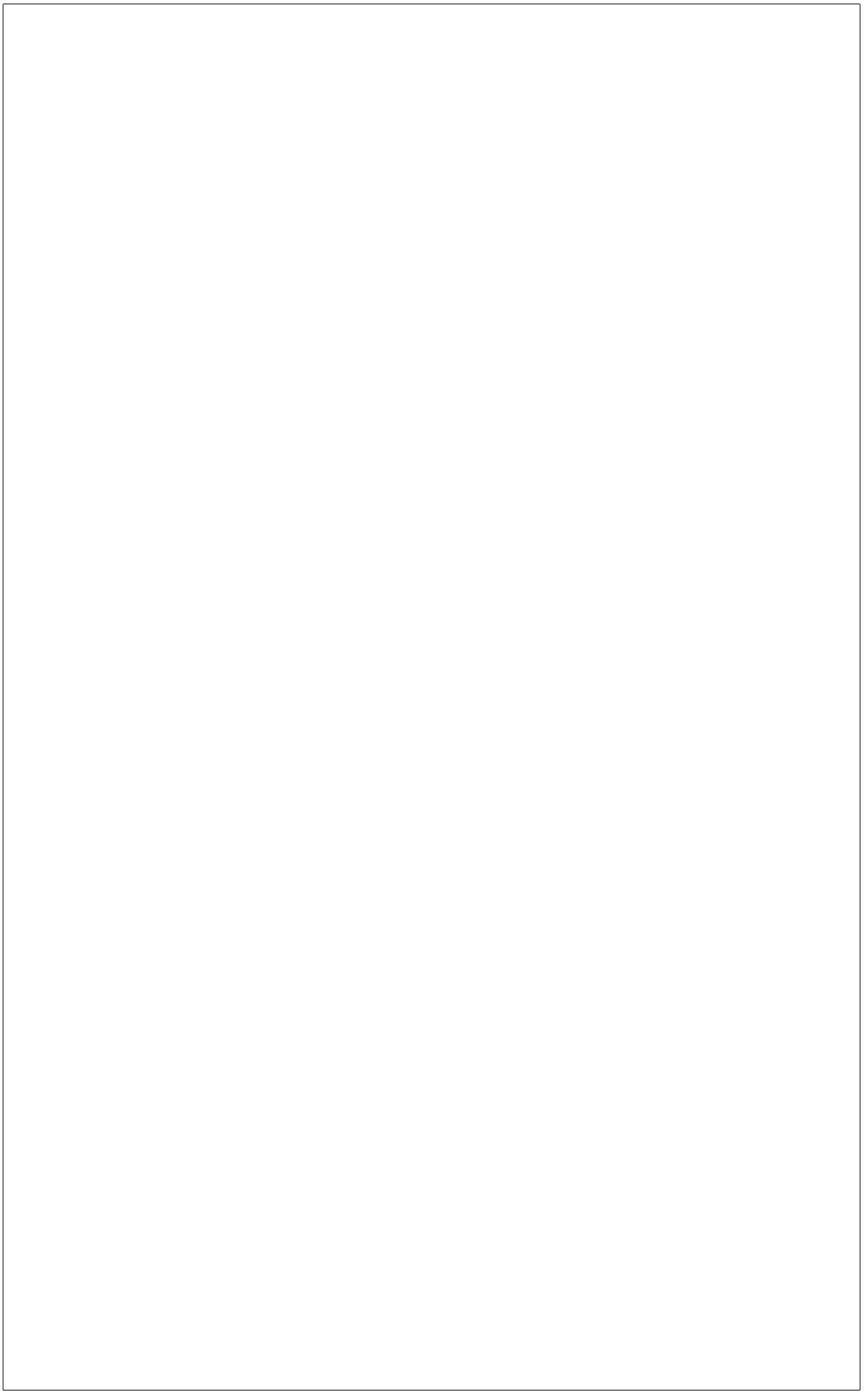
Condition (2.28a) ensures that the normal repulsion rate is larger than the tangential stretch due to shear strain along the LCS, in accordance with definition 5. Conditions (2.28c) and (2.28d) suffice to enforce that the normal repulsion rate attains a local extremum along the LCS, relative to all nearby material surfaces. Lastly, condition (2.28b) ensures that this is a strict local maximum.

From condition (2.28c) and the orthormality of the Cauchy-Green strain eigenvectors (cf. equation (2.25)), it follows that any initial LCS surface is tangent to the planes locally spanned by $\boldsymbol{\xi}_1(\mathbf{x}_0)$ and $\boldsymbol{\xi}_2(\mathbf{x}_0)$. Thus, an integral curve of any (normalized) linear combination of the $\boldsymbol{\xi}_1$- and $\boldsymbol{\xi}_2$-direction fields, launched from an arbitrary point of the surface $\mathcal{M}(t_0)$, will lever leave $\mathcal{M}(t_0)$. Hence:

Remark 1 *(Invariance of initial positions of repelling LCSs).*
The initial position $\mathcal{M}(t_0)$ of any repelling LCS (definition 6) is an invariant manifold of the autonomous dynamical system

$$\mathbf{r}' = a\boldsymbol{\xi}_1(\mathbf{r}) + b\boldsymbol{\xi}_2(\mathbf{r}), \quad a^2 + b^2 = 1. \tag{2.31}$$

Note that the converse of remark 1 does not hold. That is, a material surface $\Xi(t_0)$ which is an invariant manifold of all (normalized) linear combinations of $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ does not necessarily correspond to a repelling LCS $\mathcal{M}(t_0)$ — that is, unless $\Xi(t_0)$ also satisfies conditions (2.28a), (2.28b) and (2.28d).



3 Method

3.1 The considered flow systems

3.1.1 *Flow systems defined by analytical velocity fields*

Steady Arnold-Beltrami-Childress flow

The Arnold-Beltrami-Childress (henceforth abbreviated to ABC) flow is a three-dimensional, incompressible velocity field which solves the Euler equations exactly. It is a simple example of a fluid flow which can exhibit chaotic behaviour (Frisch 1995, p.204). In terms of the Cartesian coordinate vector $\mathbf{x} = (x, y, z)$, the system can be expressed mathematically as

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}) = \begin{pmatrix} A\sin(z) + C\cos(y) \\ B\sin(x) + A\cos(z) \\ C\sin(y) + B\cos(x) \end{pmatrix}, \tag{3.1}$$

where A, B and C are parameters which dictate the nature of the flow pattern. The inherent periodicity with regards to the Cartesian axes naturally leads to a domain of interest $\mathcal{U} = [0,\ 2\pi]^3$, with periodic boundary conditions imposed in x, y and z.

Here, the parameter values

$$A = \sqrt{3}, \quad B = \sqrt{2}, \quad C = 1 \tag{3.2}$$

were used, as has been common in litterature (e.g. by Oettinger and Haller (2016)), as these values are known to result in chaotic tracer trajectories (Zhao et al. 1993).

Unsteady Arnold-Beltrami-Childress flow

Inspired by Oettinger and Haller (2016), a temporally aperiodic modification of the ABC flow (equation (3.1)) was made by the replacements

$$\begin{aligned} B \to \widetilde{B}(t) &= B + B \cdot k_0 \tanh(k_1 t)\cos\big((k_2 t)^2\big), \\ C \to \widetilde{C}(t) &= C + C \cdot k_0 \tanh(k_1 t)\sin\big((k_3 t)^2\big), \end{aligned} \tag{3.3}$$

with A, B and C given by equation (3.2), where the parameters values

$$k_0 = 0.3, \quad k_1 = 0.5, \quad k_2 = 1.5, \quad k_3 = 1.8, \tag{3.4}$$

were used. The time dependence of the $\widetilde{B}$ and $\widetilde{C}$ coefficients is illustrated in figure 3.1.

3.1.2 *Flow systems defined by gridded velocity data*

Oceanic currents in the Førde Fjord

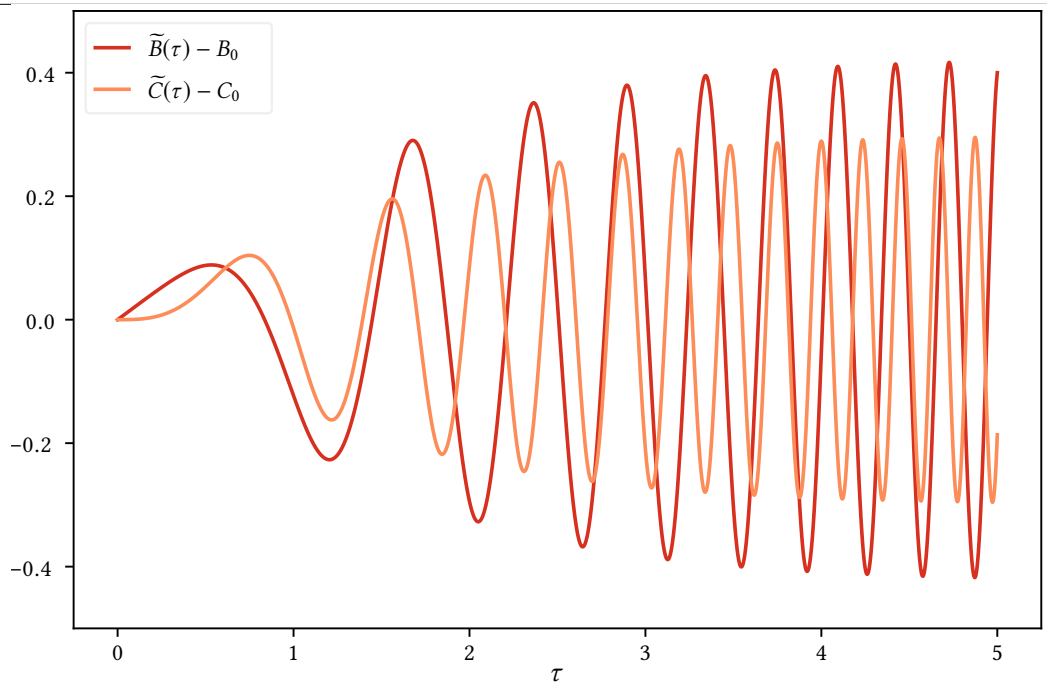


Figure 3.1: Time dependence of the coefficient functions for unsteady ABC flow, defined in equations (3.2)–(3.4).

Motiver med potensielt sjødeponi for gruveavfall

In order to identify LCSs in three-dimensional flow by means of geodesic level set approximations, a system which has been studied extensively in the literature was chosen. The system is a simple example of a fluid flow which can exhibit chaotic behaviour (Frisch 1995, p.204).

3.2 Computing the flow map and its directional derivatives

3.2.1 *Advecting a set of tracers*

The variational framework for computing LCSs is based upon the advection of non-interacting tracers, as described in section 2.2, by the systems mentioned in section 3.1. The computational domains $\mathcal{U}$ were discretized by a set equidistant tracers, effectively creating a uniform grid with tracers placed on and within the domain boundaries of $\mathcal{U}$. The grid parameters are summarized in table 3.1.

In order to increase the precision of the computed Cauchy-Green strain tensor field, it is necessary to increase the accuracy with which one computes the Jacobian of the flow map, as their accuracies are intrinsically linked; which follows from equation (2.24). Accordingly, the flow map Jacobian was computed directly, by means of simultaneously solving the

Table 3.1: Grid parameters for advection in the considered flow systems. Note that, for the fjord system, the domain extents and grid spacings are given in units of metre.

	Analytical ABC flow	Fjord model data
Computational domain	$[0,\ 2\pi]^3$	$[0,\ 500] \times [0,\ 500] \times [50,\ 300]$
N_x, N_y, N_z	256, 256, 256	200, 200, 100
$\Delta x = \Delta y = \Delta z$	$2.5 \cdot 10^{-2}$	2.5

twelve coupled ODEs given by equations (2.16) and (2.22) while letting the underlying velocity field transport the tracers. All twelve ODEs were solved simultaneously, via the Dormand-Prince 8(7) method (see section 2.1.1 and, in particular, table 2.3). The dynamic step length adjustment procedure will be outlined in detail in section 3.2.2.

In this framework, the tracer advection takes second stage to the 'advection' of the components of the flow map Jacobian. As it turns out, the increase in mathematical complexity which the coupling terms introduces is a small price to pay for the increased precision compared to the straightforward approach of applying a finite diffference scheme to the advected flow map (Oettinger and Haller 2016). This is also evident from previous 'finite difference-based' LCS computing endeavors, in which the use of several grids of tracers was necessitated in order to accurately compute flow map Jacobian (Løken 2017; Farazmand and Haller 2012a).

3.2.2 *The implementation of dynamic Runge-Kutta step size*

In order to implement automatic step size control, the procedure suggested by Hairer, Nørsett, and Wanner (1993, pp.167–168) was followed closely. A starting step size h needs to be prescribed; this generally differs based upon the (pseudo-)time scale of the underlying system. For the first solution step, the embedded Dormand-Prince 8(7) method, as described in section 2.1.1 and table 2.3, yields the two approximations x_1 and $\widehat{x}_1$, from which the difference $x_1 - \widehat{x}_1$ can be used as an estimate of the error of the less precise result. The idea is to enforce the error of the numerical solution to satisfy, componentwise:

$$\left|x_{1,i} - \widehat{x}_{1,i}\right| \leq \mathrm{sc}_i, \quad \mathrm{sc}_i = \mathrm{Atol}_i + \max\left(\left|x_{1,i}\right|, \left|\widehat{x}_{1,i}\right|\right) \cdot \mathrm{Rtol}_i, \tag{3.5}$$

where Atol_i and Rtol_i are the desired absolute and relative tolerances. For this project, the tolerance values

$$\mathrm{Atol}_i = 10^{-7}, \quad \mathrm{Rtol}_i = 10^{-7} \tag{3.6}$$

were used throughout.

As a measure of the numerical error,

$$\mathrm{err} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_{1,i} - \widehat{x}_{1,i}}{\mathrm{sc}_i}\right)^2} \tag{3.7}$$

is used. Then, err is compared to unity in order to find an optimal step size. From definition 2, it follows that err scales like h^{q+1}, where $q = \min(p, \widehat{p})$. Under the assumption $1 \approx K h_{\mathrm{opt}}^{q+1}$,

one finds the optimal step size from

$$h_{\text{opt}} = h \cdot \left(\frac{1}{\text{err}}\right)^{\frac{1}{q+1}}. \tag{3.8}$$

If $\text{err} \leq 1$, the suggested solution step is accepted, the (pseudo-)time variable t is increased by h, and the step length is modified according to equations (3.8) and (3.9). Which of the two approximations x_{n+1} or $\widehat{x}_{n+1}$ is used to continue the integration generally depends on the embedded Runge-Kutta method in question. Continuing the integration with the higher order result is commonly referred to as *local extrapolation*. The Dormand-Prince 8(7) method is tuned in order to minimize the order of the higher order result; accordingly, local extrapolation was used throughout. If $\text{err} > 1$, the solution step is rejected, and the step length decreased before attempting another step. The procedure for updating the time step can be summarized as follows:

$$h_{\text{new}} = \begin{cases} \min(\text{fac}_{\max} \cdot h, \text{fac} \cdot h_{\text{opt}}) & \text{if the solution step is accepted,} \\ \text{fac} \cdot h_{\text{opt}}, & \text{if the solution step is rejected,} \end{cases} \tag{3.9}$$

where fac and $\text{fac}_{\max}$ are numerical safety factors, intended to prevent increasing the step size *too* much. Here, the parameter values

$$\text{fac} = 0.8, \quad \text{fac}_{\max} = 2.0, \tag{3.10}$$

were used throughout.

3.3 Computing Cauchy-Green strain eigenvalues and -vectors

Computing the Cauchy-Green strain tensor field directly, by performing a series of matrix products per its definition in equation (2.24), and then solving for its eigenvalues and -vectors turns out to be numerically disadvantageous (Oettinger and Haller 2016). In particular, this method leaves the smallest eigenvalues quite susceptible to numerical round-off error. A fully equivalent, more numerically sound way of identifying the Cauchy-Green strain eigenvalues and -vectors is based on performing an SVD decomposition of the Jacobian field of the flow map, i.e.,

$$\nabla \boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{3.11}$$

where the asterisk refers to the adjoint operation, $\mathbf{U}$ and $\mathbf{V}$ are square, unitary matrices and $\boldsymbol{\Sigma}$ is a square, diagonal matrix with nonnegative real numbers — the *singular values* of $\nabla \boldsymbol{\phi}$ — on the diagonal. Because the flow map Jacobian is real-valued, so too are the matrices $\mathbf{U}$ and $\mathbf{V}$. The eigenvalues of the right Cauchy-Green strain tensor (cf. equations (2.24) and (2.25)) are given by the squares of the singular values, that is, $\lambda_i(\mathbf{x}_0) = \left(\sigma_i(\mathbf{x}_0)\right)^2$, and the corresponding orthonormal eigenvectors are found in the columns of $\mathbf{V}$.

Interpolating the Cauchy-Green strain eigenvalues

For computing LCSs, the Cauchy-Green strain eigenvalues frequently need to be evaluated inbetween the grid points. Moreover, as suggested by the existence criterion given in

equation (2.28b), all of the second derivatives of $\lambda_3(\mathbf{x}_0)$ are also needed. Accordingly, the eigenvalues were interpolated by means of cubic trivariate B-splines, in order to ensure continuous second derivatives. For this purpose, the Bspline-Fortran library written by Williams (2018), was used.

Interpolating the Cauchy-Green strain eigenvectors

Just like the eigenvalues of the Cauchy-Green strain tensor field, its eigenvectors frequently need to be evaluated inbetween the grid points in order to compute LCSs. Like the strain eigenvalues, the strain eigenvectors were interpolated by means of cubic trivariate B-splines; though with a twist, in order to remove local orientational discontinuities. In particular, the local stretch is equal in magnitude along the negative ξ_3 axis as that of its positive counterpart, and there is no a priori reason to expect the SVD decomposition (cf. equation (3.11)) to follow any particular convention regarding the 'sign' of the computed eigenvectors.

The interpolation routine outlined here is a generalization of a similar special-purpose linear interpolation routine which has previously been utilized to compute LCSs in two spatial dimensions (Onu, Huhn, and Haller 2015; Løken 2017). The routine is based upon careful monitoring and local reorientation prior to cubic interpolation, and its two-dimensional equivalent is illustrated in ***SETT INN REF*** — the principles are similar in three dimensions, but illustrating a two-dimensional projection of the three-dimensional case easily became horribly cluttered.

First, the 64 (in two dimensions: 16) nearest neighboring grid points to any given coordinate $\mathbf{r}$ are identified. Choosing a vector at a corner of this local interpolation voxel, orientational discontinuities inbetween the grid elements are found by inspecting the inner products of the ξ_i vectors of the remaining grid points with the pivot. Rotations exceeding 90^{deg} are labelled as orientational discontinuities, and are corrected by flipping the corresponding vectors by 180^{deg}. For each of ξ_i's three components, cubic B-spline interpolation is used within the interpolation voxel in order to find $\xi_i(\mathbf{r})$, which is then normalized, like the ξ_i vectors defined at the grid points are a priori.

3.4 The method of geodesic level sets

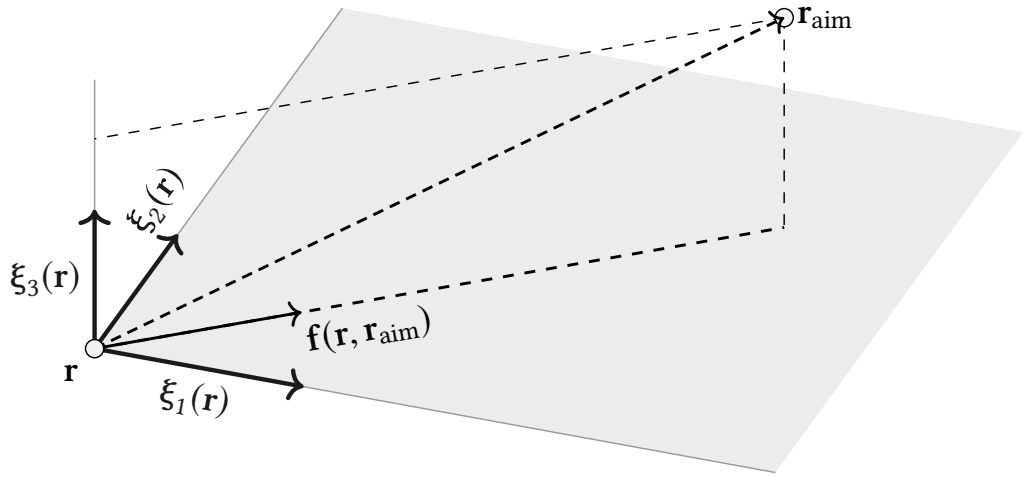


Figure 3.2: Geometrix interpretation of the eigenvectors of the Cauchy-Green strain tensor. The central unit cell is stretched and deformed under the flow map $\boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0)$. The local stretching is the largest in the direction of ξ_3, the eigenvector which corresponds to the largest eigenvalue, λ_3, of the Cauchy-Green strain tensor, defined in equation (2.25). Along the ξ_i axes, the stretch factors are given by $\sqrt{\lambda_i}$, respectively.

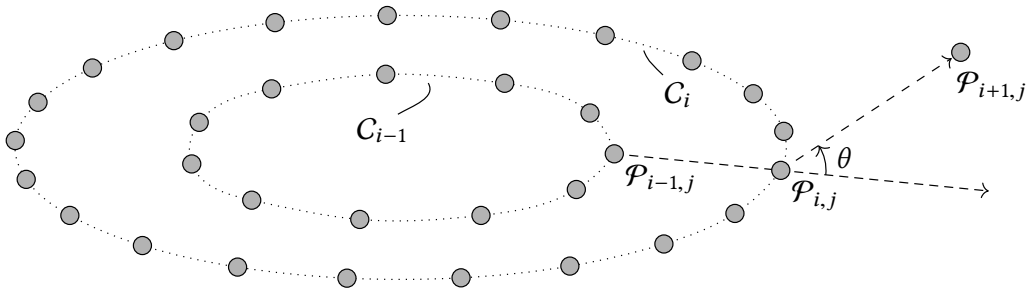
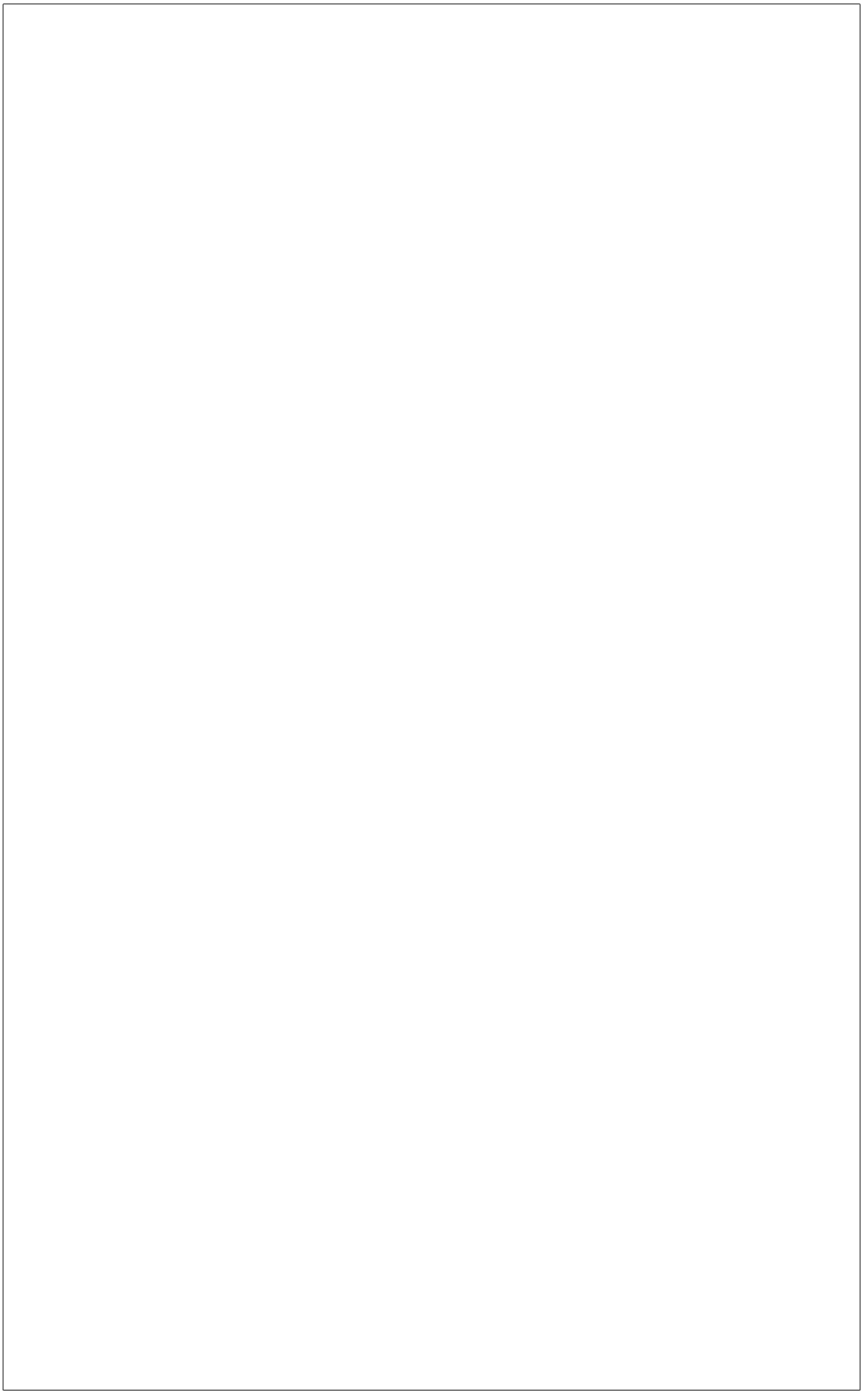
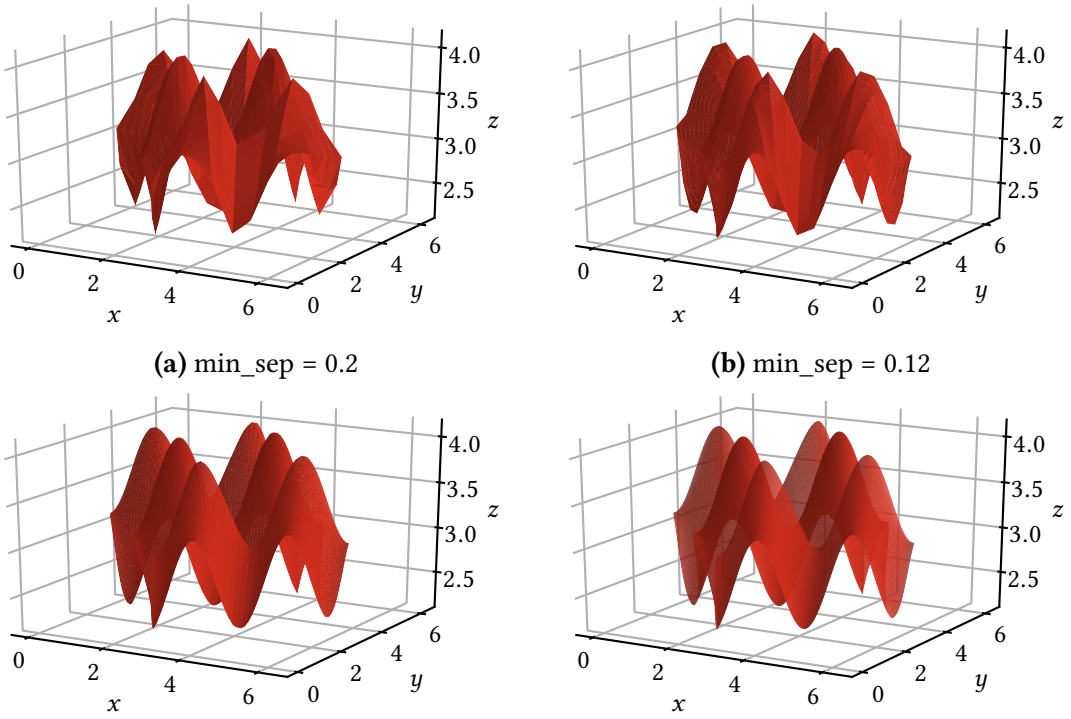


Figure 3.3: Geometrix interpretation of the eigenvectors of the Cauchy-Green strain tensor. The central unit cell is stretched and deformed under the flow map $\boldsymbol{\phi}_{t_0}^{t}(\mathbf{x}_0)$. The local stretching is the largest in the direction of ξ_3, the eigenvector which corresponds to the largest eigenvalue, λ_3, of the Cauchy-Green strain tensor, defined in equation (2.25). Along the ξ_i axes, the stretch factors are given by $\sqrt{\lambda_i}$, respectively.

- ABC-strømning (begge former), fjordstrømning (vist v/ kartprojeksjon(er)), oppsett av grids og beregning av λ, ξ v/ SVD
- Interpolasjon av skalare- og vektorstørrelser
- Fullstendig beskrivelse av opprinnelig metode (lett modifisert GLS; tanvec, prevvec, levelsets etc.), muligens med flowchart
- Vis/beskriv eksplisitt noen av problemene/utfordringene med opprinnelig metode
    - Veldig mange frihetsgrader
    - (Tidvis) tilfeldighetspreget oppførsel i regioner med rasktvarierende ξ_3, hvor godt vi traff avh. av hvilket vinkeloffset vi traff med først (opp eller ned)
    - Langsom metode; Trenger gjerne mange forsøk om det først går galt
- Utfyllende meskrivelse av ny metode, med hovedfokus på forskjellene fra opprinnelig versjon. Flowchart?
    - Kan tillate os dette fordi vi har en ekstra frihetsgrad sml med system fra levelset-paper
    - Utvikling langs «strains» er en tredimensjonal videreføring av strainlines (siter Løken/Nordgreen 2017; F&H 2012)
    - Mer konsekvent oppførsel nær sterke diskontinuiteter, god konvergens
- Nytt av versjon 2 er kontinuerlige self-intersection-checks
    - Beskriv bruk av Möller-Trumbore osv.
- Seleksjon av lokalt sterkest repulsive materialoverflater v/ lokal sjekk
    - Sjekk hvert punkt på mangfoldigheten ift ABD
    - Behold også punkt som ikke er i ABD, så fremt de er nærme nok minst et annet punkt i ABD
    - Filtrer vekk LCS-kandidater som er for små



4 Results



(a) min_sep = 0.2

(b) min_sep = 0.12

(c) min_sep = 0.04

(d) min_sep = 0.01

Figure 4.1: Manifolds generated from analytically determined vector field.

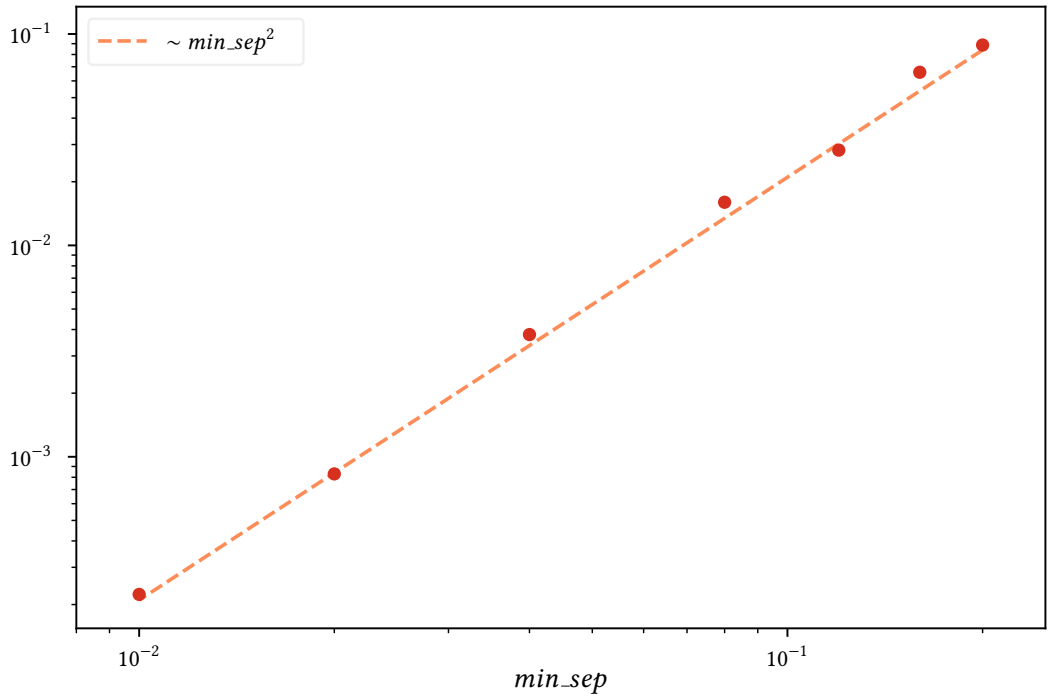
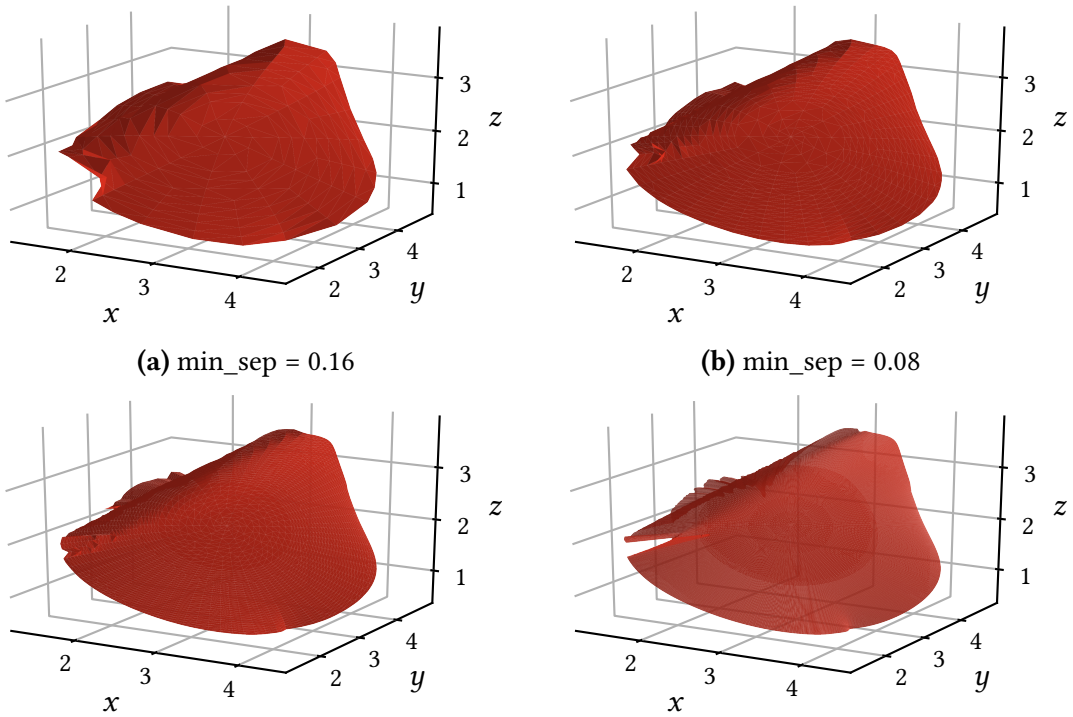


Figure 4.2: RMS error of sinusoidal manifolds



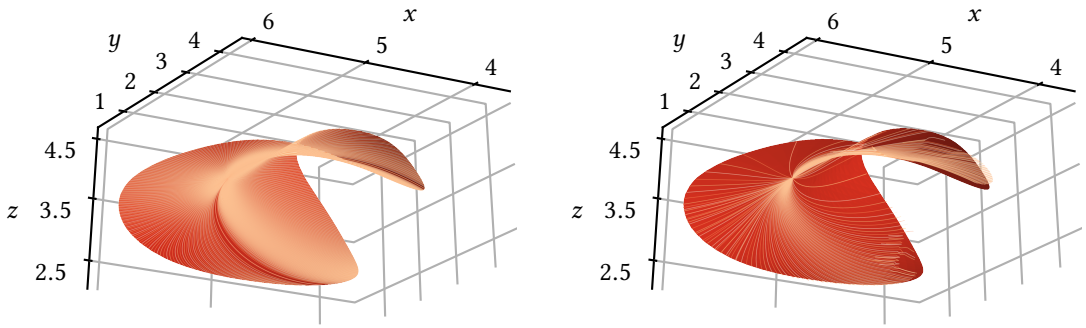
(a) min_sep = 0.16

(b) min_sep = 0.08

(c) min_sep = 0.04

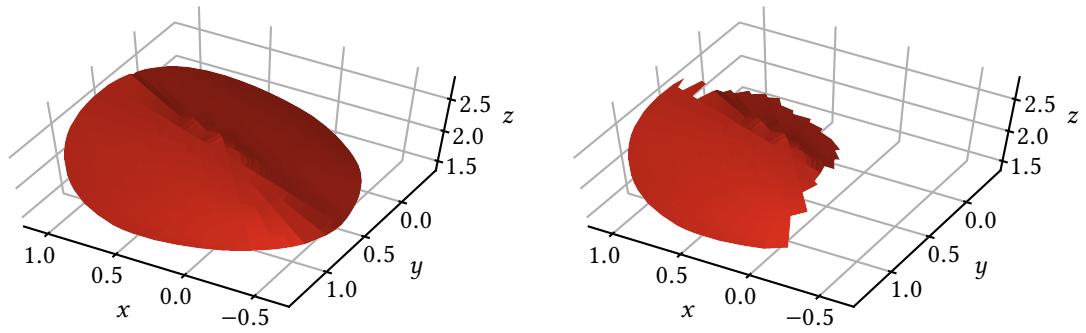
(d) min_sep = 0.01

Figure 4.3: Numerical convergence as min_sep decreases



(a) Trajectories forced radially outwards **(b)** Pure linear combinations of ξ_1 and ξ_2

Figure 4.4: Trajectories with local directionality given by linear cominations of ξ_1 and ξ_2.



(a) Manifold **(b)** LCS candidate

Figure 4.5: Converting a manifold to an LCS candidate

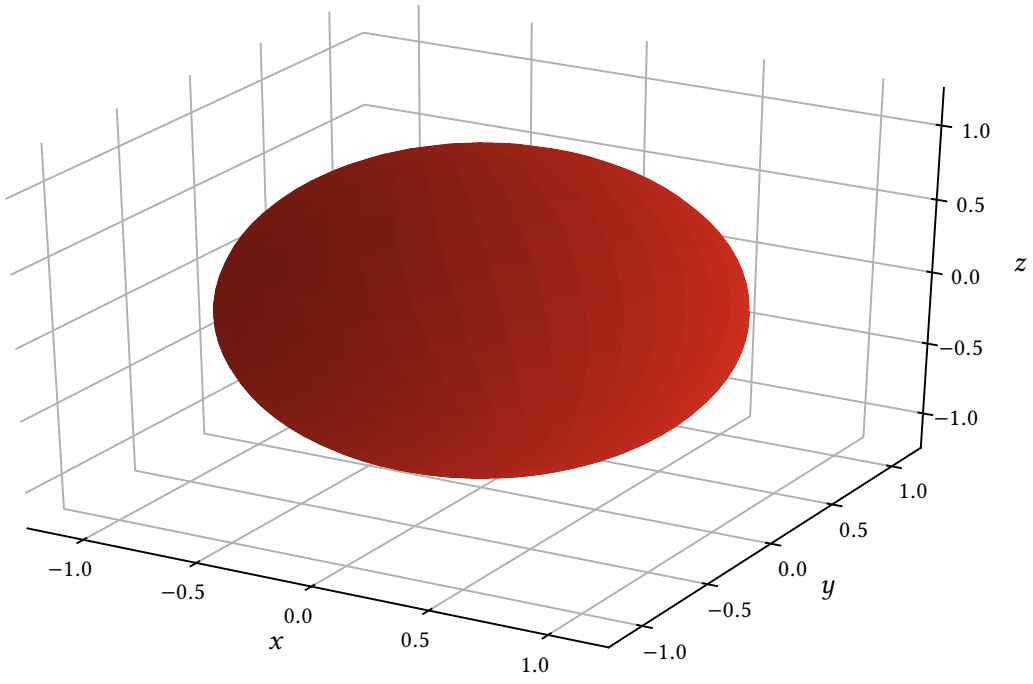


Figure 4.6: The identified LCS for the spherical flow

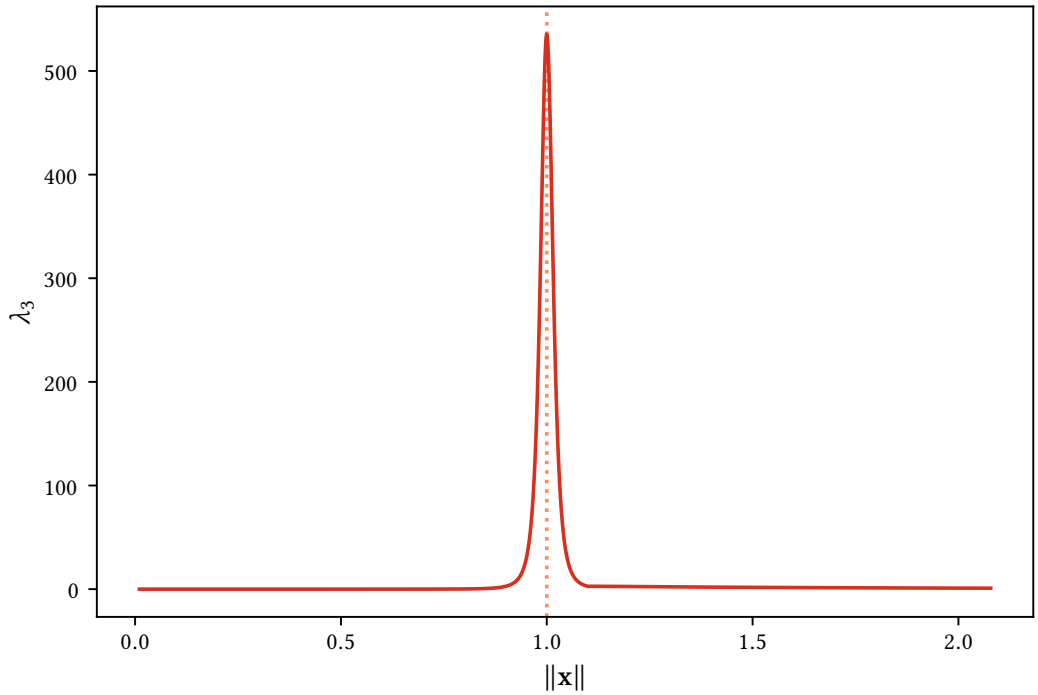
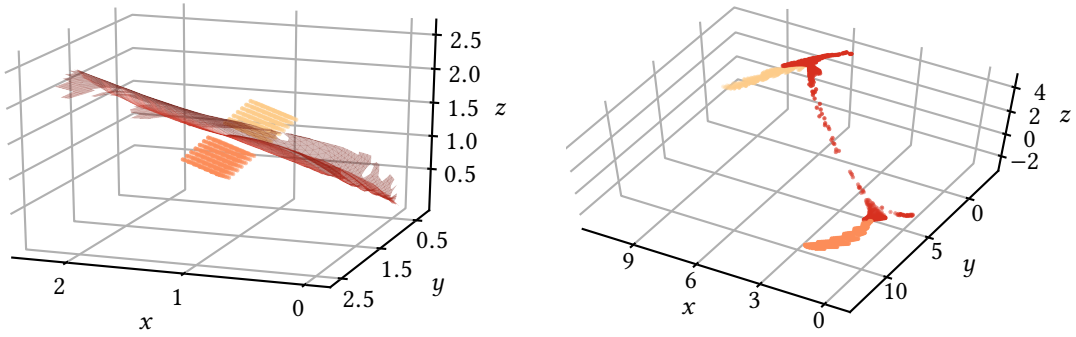
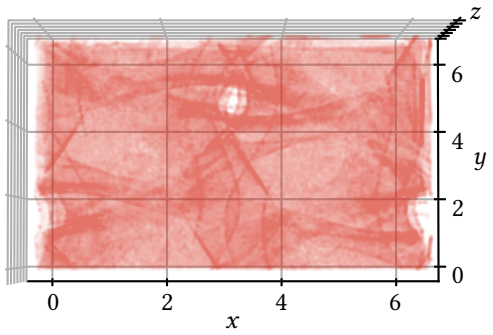


Figure 4.7: λ_3 as a function of radius, for the spherical flow. Displayed here is the arithmetic average across all solid angles.

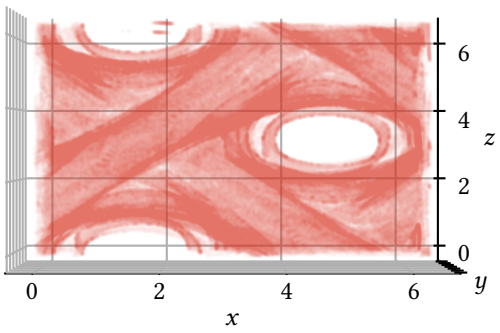


(a) $t = 0$ **(b)** $t = 5$

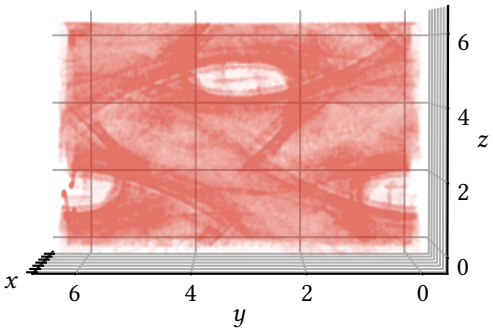
Figure 4.8: Blob test for verifying expected LCS behaviour



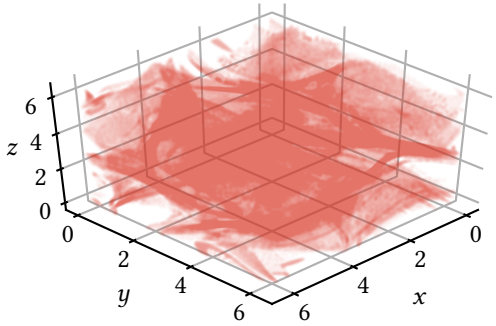
(a) View along the negative z-axis



(b) View along the positive y-axis

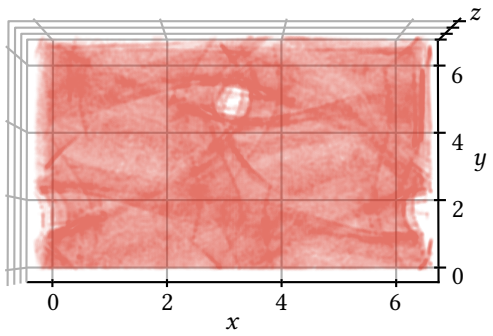


(c) View along the positive x-axis

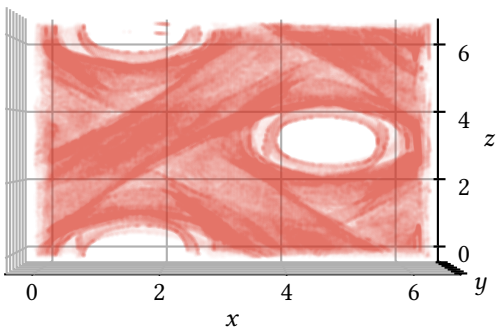


(d) Approximately isometric view

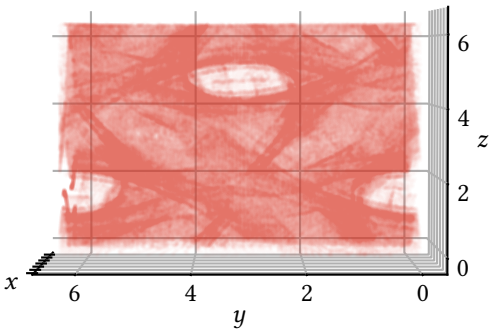
Figure 4.9: ABD domain obtained for stationary flow. $\epsilon = 0.005$.



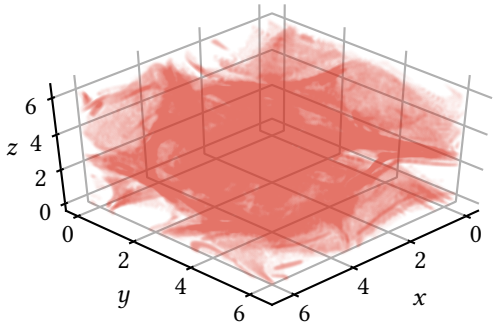
(a) View along the negative z-axis



(b) View along the positive y-axis

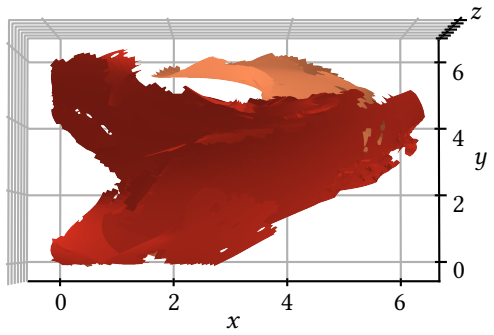


(c) View along the positive x-axis

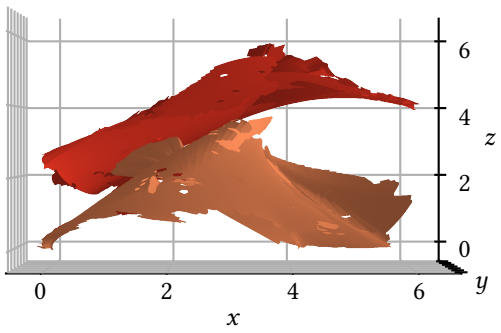


(d) Approximately isometric view

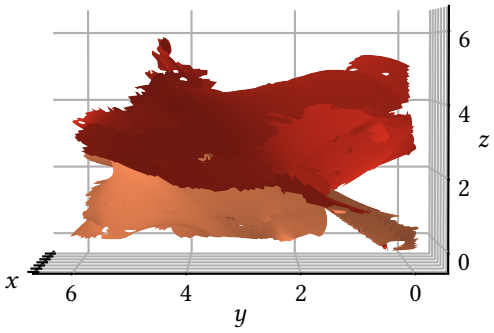
Figure 4.10: ABD domain obtained for dynamic flow. $\epsilon = 0.005$.



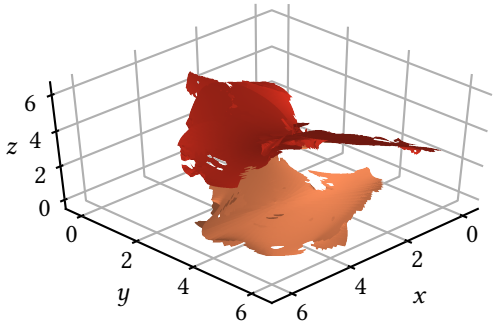
(a) View along the negative z-axis



(b) View along the positive y-axis

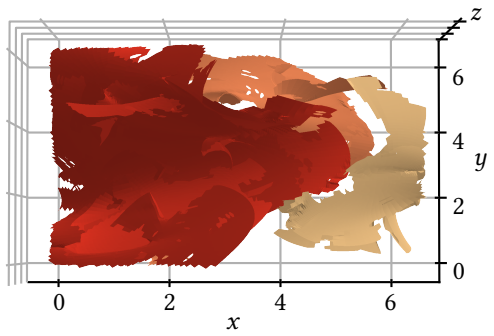


(c) View along the positive x-axis

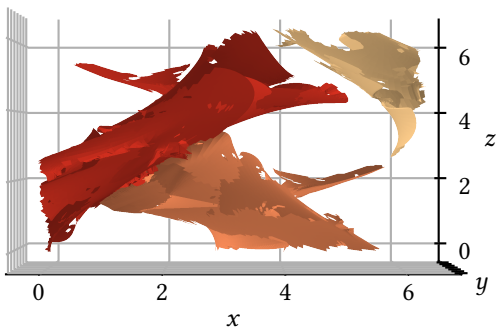


(d) Approximately isometric view

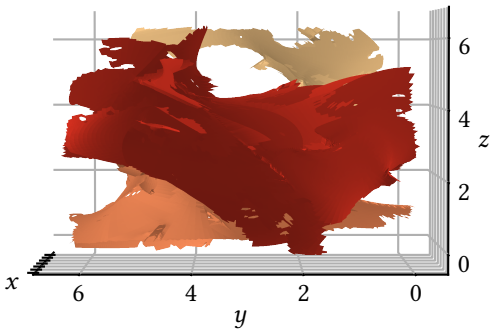
Figure 4.11: LCSs obtained for stationary ABC flow



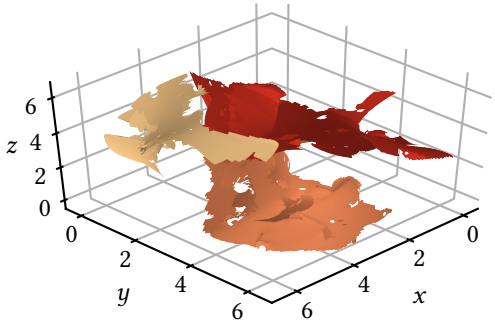
(a) View along the negative z-axis



(b) View along the positive y-axis

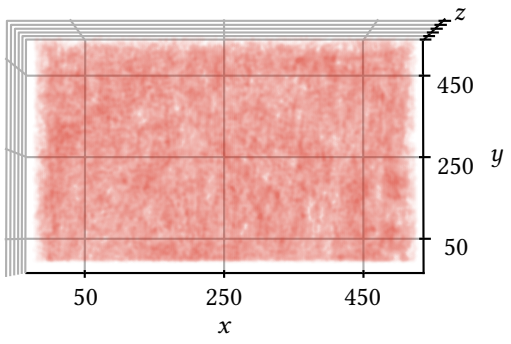


(c) View along the positive x-axis

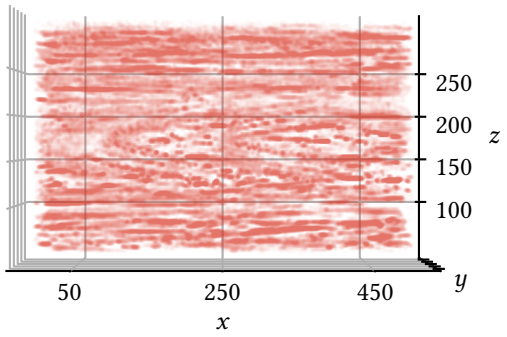


(d) Approximately isometric view

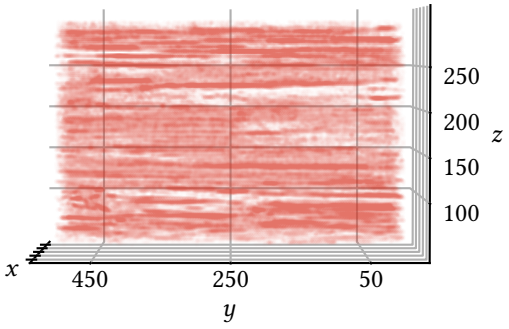
Figure 4.12: LCSs obtained for dynamic ABC flow



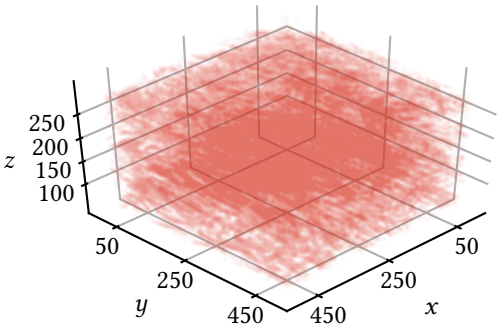
(a) View along the negative z-axis



(b) View along the positive y-axis

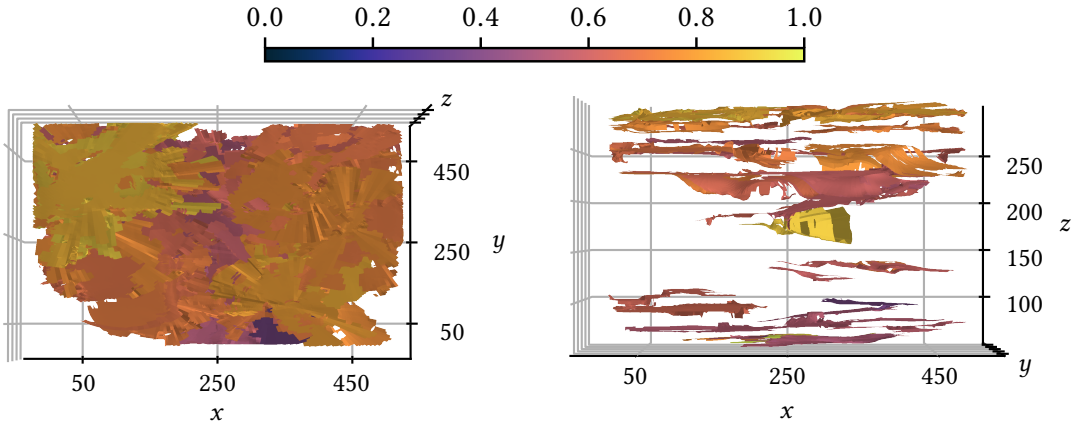


(c) View along the positive x-axis



(d) Approximately isometric view

Figure 4.13: ABD domain obtained for gridded data. The axes have dimension meters. z = 0 corresponds to the sea level, increasing downwards. $\epsilon = 1$.



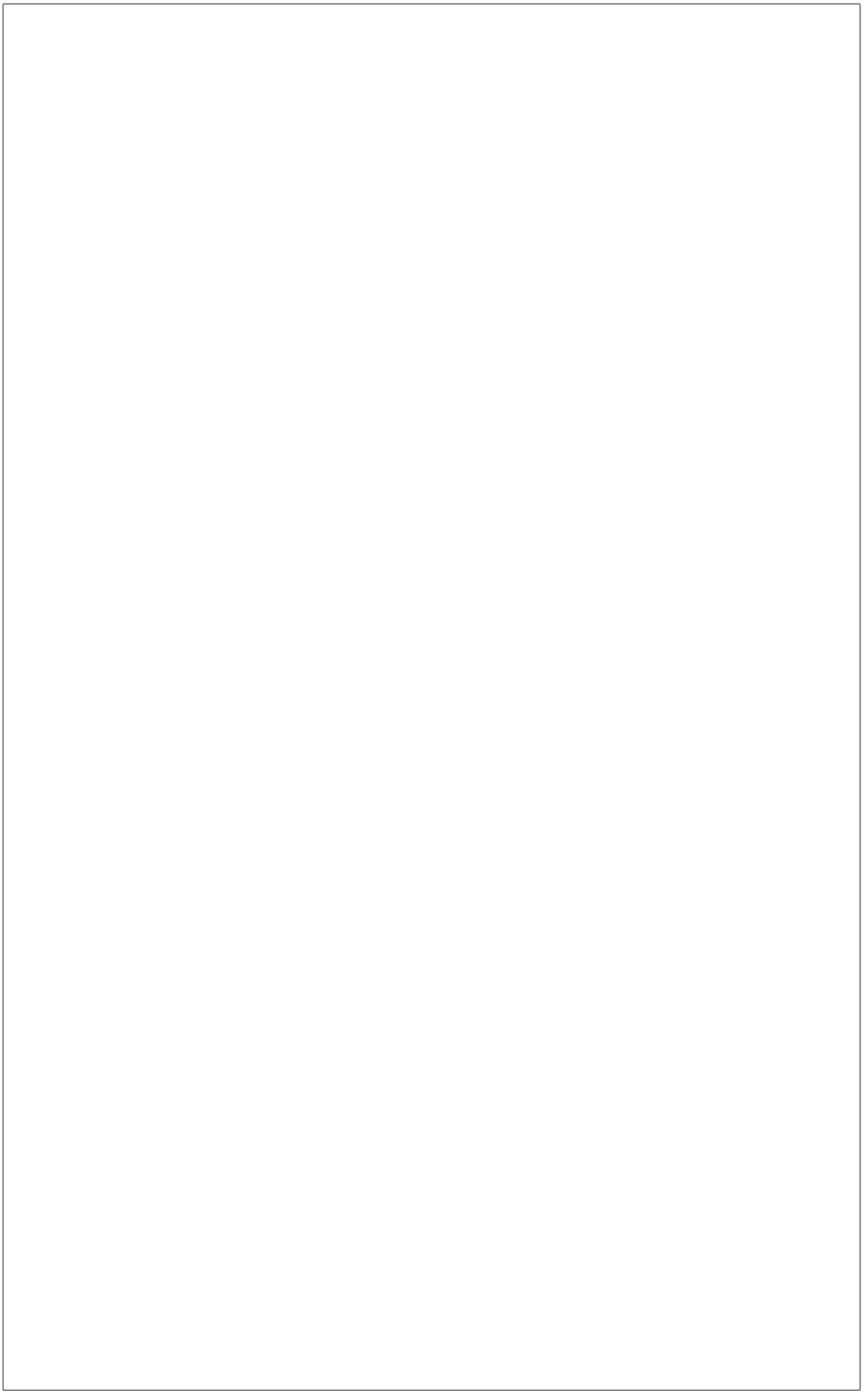
(a) View along the negative z-axis

(b) View along the positive y-axis

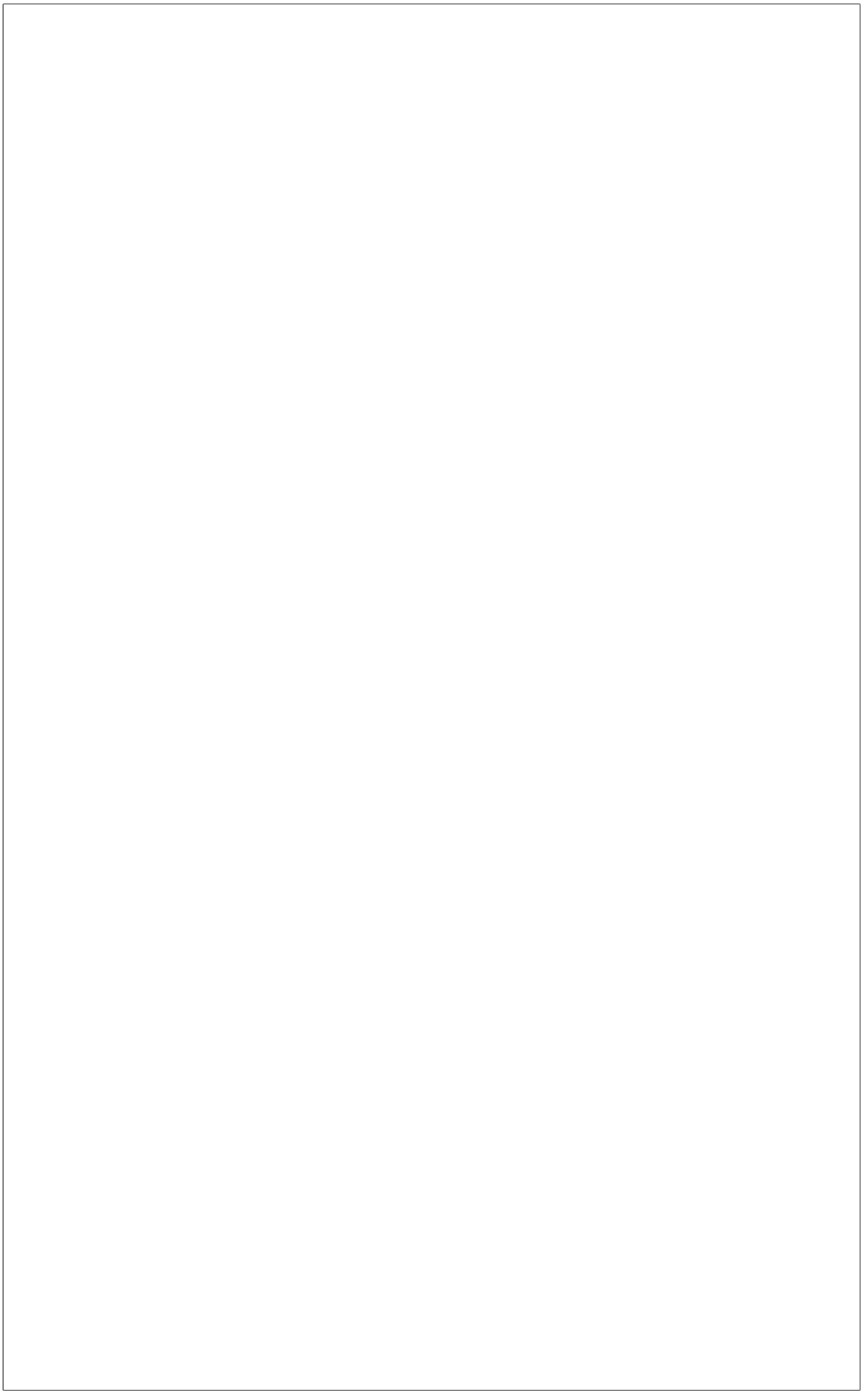
(c) View along the positive x-axis

(d) Approximately isometric view

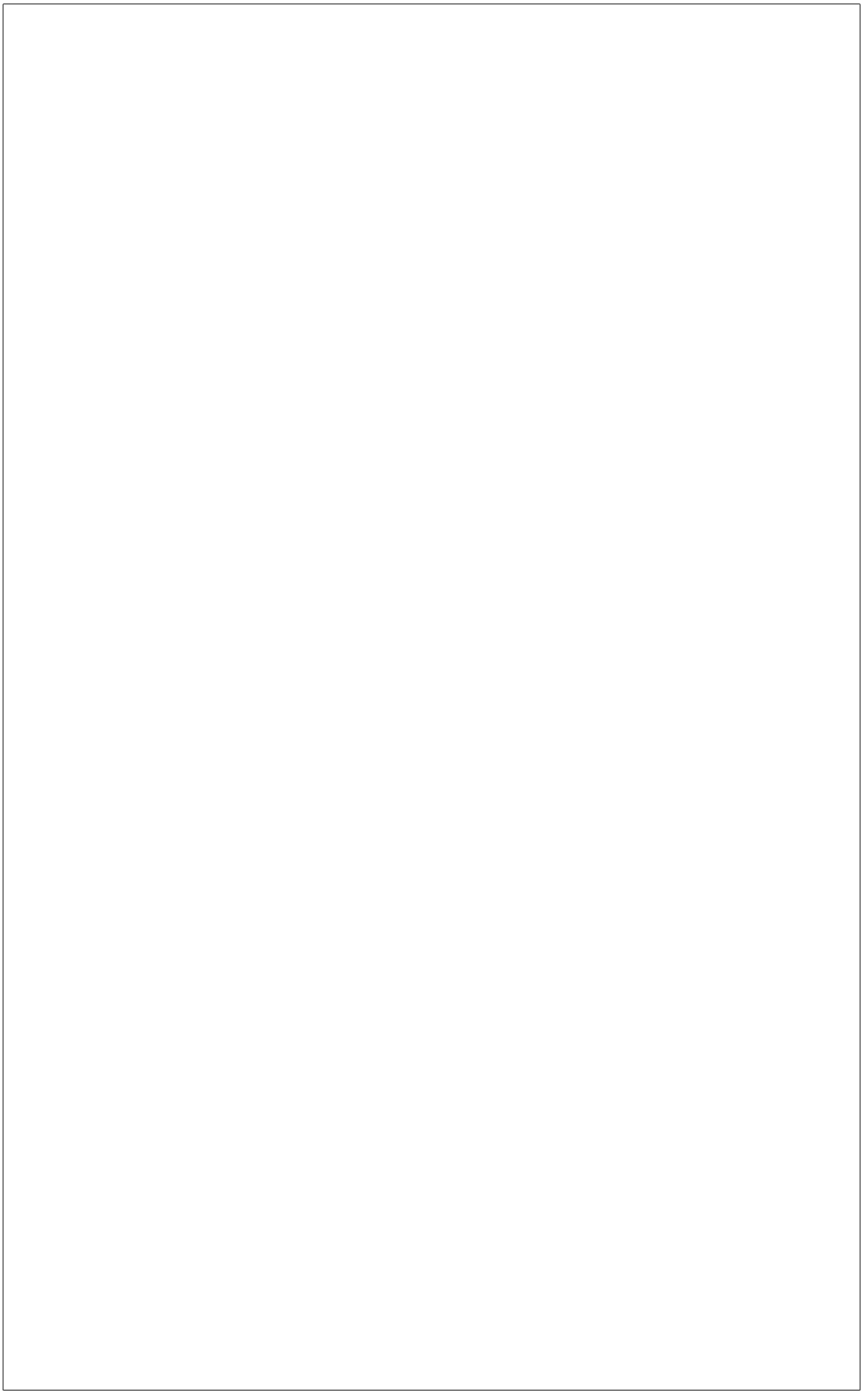
Figure 4.14: LCSs obtained for stationary ABC flow



5 Discussion



6 Conclusions



References

de Boor, C. (1978). *A Practical Guide to Splines*. 1st ed. Springer-Verlag New York. ISBN: 0-387-95366-3.

Farazmand, M. and Haller, G. (2012a). "Computing Lagrangian coherent structures from their variational theory". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1, p. 013128. ISSN: 1054-1500. DOI: 10.1063/1.3690153.

Farazmand, M. and Haller, G. (2012b). "Erratum and addendum to 'A variational theory of hyperbolic Lagrangian coherent structures' [Physica D 240 (2011) 547–598]". In: *Physica D: Nonlinear Phenomena* 241.4, pp. 439–441. ISSN: 0167-2789. DOI: 10.1016/j.physd.2011.09.013.

Frisch, U. (1995). *Turbulence, the legacy of A.N. Kolmogorov*. 1st ed. Cambridge University Press. ISBN: 0-521-45103-5.

Hairer, E., Nørsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*. 2nd ed. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-56670-0. Corrected 3rd printing, 2008.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. 2nd ed. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-05221-7. Corrected 2nd printing, 2002.

Haller, G. (2010). "A variational theory of hyperbolic Lagrangian Coherent Structures". In: *Physica D: Nonlinear Phenomena* 240.7, pp. 547–598. ISSN: 0167-2789. DOI: 10.1016/j.physd.2010.11.010.

Karrasch, D. (2012). "Comment on 'A variational theory of hyperbolic Lagrangian coherent structures, Physica D 240 (2011) 574–598'". In: *Physica D: Nonlinear Phenomena* 241.17, pp. 1470–1473. ISSN: 0167-2789. DOI: 10.1016/j.physd.2012.05.008.

Løken, A. M. T. (2017). "Sensitivity to Numerical Integration Scheme in Calculation of Lagrangian Coherent Structures". Unpublished specialization project. Available at http://folk.ntnu.no/amloken/sensitivity_to_numerical_integration_scheme_loken.pdf.

Oettinger, D. and Haller, G. (2016). "An autonomous dynamical system captures all LCSs in three-dimensional unsteady flows". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 26.10. ISSN: 1054-1500. DOI: 10.1063/1.4965026.

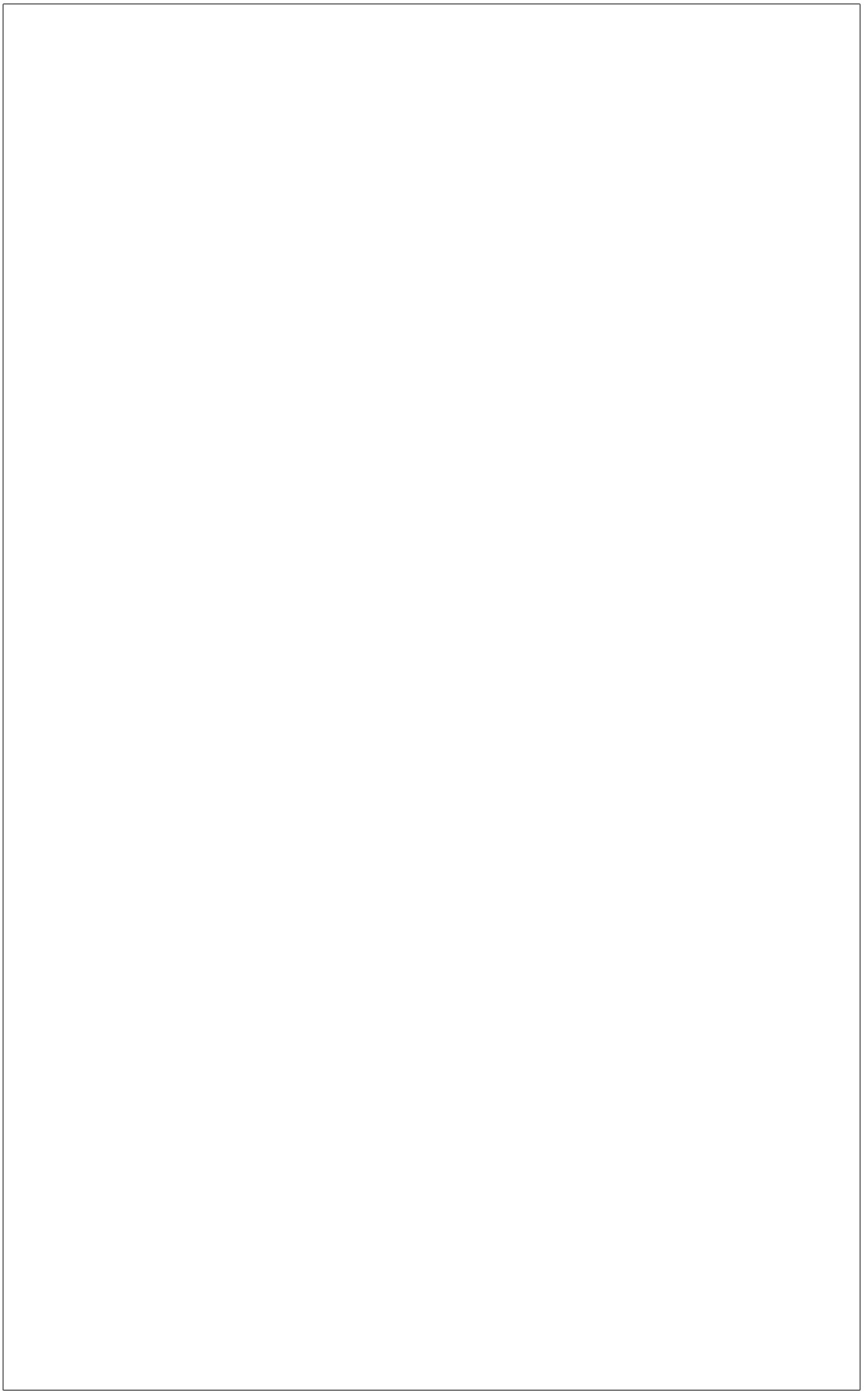
Onu, K., Huhn, F., and Haller, G. (2015). "LCS Tool: A computational platform for Lagrangian coherent structures". In: *Journal of Computational Science* 7, pp. 26–36. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2014.12.002.

Prince, P. and Dormand, J. (1981). "High order embedded Runge-Kutta formulae". In: *Journal of Computational and Applied Mathematics* 7.1, pp. 67–75. ISSN: 0377-0427. DOI: 10.1016/0771-050X(81)90010-3.

Stoer, J. and Bulirsch, R. (2002). *Introduction to Numerical Analysis*. Springer-Verlag New York. ISBN: 978-1-4419-3006-4.

Williams, J. (2018). *Bspline-Fortran: Multidimensional B-Spline Interpolation of Data on a Regular Grid*. Zenodo. DOI: 10.5281/zenodo.1215290. Open Access Software.

Zhao, X.-H. et al. (1993). "Chaotic and Resonant Streamlines in the ABC flow". In: *SIAM Journal on Applied Mathematics* 53.1, pp. 71–77. DOI: 10.1137/0153005.



A Appendix A