[21] C. M. Vest and M. L. Lawson. Onset of convection near a suddenly heated horizontal wire. *International Journal of Heat and Mass Transfer*, 15(6):1281–1283, 1972.

[22] D. B. White. The planforms and onset of convection with a temperature dependent viscosity. *Journal of Fluid Mechanics*, 191:247–286, 1988.

CHAPTER VI

Reflections and future work

Although laboratory methods like those described in this thesis obviously lack the ability to capture the full complement of behaviors that are characteristic of a complex system like the Earth's mantle, experiments will continue to play a fundamental role in identifying new fluid phenomena. A flow studied in the laboratory suffers no effects from numerical precision, numerical dissipation, or simplifying modifications made to governing equations. The laboratory flow is real and will precisely capture all the relevant physics that governs its behavior. And, when some experimental limitation becomes imprinted on the data, we often learn something new that might have otherwise been overlooked.

During the course of our experiments, a number of avenues for improvement in experimental technique have been established, and several open questions regarding the dynamics of plumes that warrant additional exploration have arisen. In the remaining discussion, these items are presented in the hope that they will prove beneficial to subsequent efforts.

Improvements to experimental technique

Our results can be greatly expanded upon by more-effectively measuring the temperature field of the flow. Thermochromic liquid crystals provide an interesting tool

for non-invasive temperature measurement, but leveraging their limited color play is a significant problem. As discussed in Chapter II, TLC's are only visible over a fraction of their advertised color play in most PIV applications. In our setup, this means we can reliably resolve the temperature field within < 1 °C of ambient. However, 1 °C represents a mere 2% of the total temperature variation in the hottest flows.

Some researchers accommodate the limited color play of TLC's by using multiple TLC formulations each having a separate active temperature band. When these multiple formulations are illuminated using monochromatic light (*e.g.*, from a laser), a series of isotherms are produced [1]. Theoretically, one could also use multiple formulations with white light thereby generating a progression of rainbow spectra instead of discrete isotherms. This multi-formulation plus white light setup may work quite well for a slow, simple flow like our laminar plumes. However for complex flows, associating a particular color with a specific temperature via image processing will be difficult. Since there are many formulations that could be simultaneously scattering the same color of light, the image processing algorithm will need to use heuristics to match each formulation with a specific location in the flow field.

The optimal approach may be to use two TLC formulations and a numerical model to reconstruct the temperature field. The two TLC formulations would be chosen such that one is visible near ambient and the other active at say half the total temperature differential. The numerical model could then be run using the PIV velocity field as input and the visible TLC contours as quality control checks.

Another major opportunity for improvement in experimental procedures is much simpler and revolves around heater design and operation. In order to quantitatively characterize observed flow dynamics and produce scalings as a function of suitable non-dimensional numbers (*e.g.*, the Rayleigh number), the operating parameters for each experiment must be well-controlled and measurable. In our experiments, we took great care to instrument the heater so that operating temperature and heat flux deliv-

ered to the syrup were known. Unfortunately, our heater design and instrumentation had limitations as described below that should be remedied.

Knowing the heat flux for all experiments is a very helpful diagnostic, and it would provide another check for temperature field reconstruction with a numerical model. To this end, each of our heaters was instrumented with a thin film heat flux gauge (Omega HFS-3), and in addition the constant temperature cases also employed a thermocouple on the syrup-wetted surface of the heater. Regrettably, the massive heat flux generated as the heater was quickly brought up to temperature in the constant temperature cases caused the heat flux gauge to fail. An alternative means of measuring heat flux instead of using the thin film sensor would be advantageous.

More importantly than measuring the heat flux, however, is ensuring that heat is only delivered to the syrup along intended and known pathways. In our setup, the tank geometry near the heater did not sufficiently limit heat loss to the tank bottom. Roughly 25% of our total heater power went into the tank bottom instead of the syrup. The issue here is that the tank bottom is also in contact with the syrup, and hence an unmeasured quantity of heat can be transferred to the syrup via the tank structure instead of directly from the heater. If experimental results are to be used as inputs or validation cases for numerical models, then the heater state becomes a critical boundary condition.

In terms of the PIV technique itself, there are two opportunities that can be readily exploited to enhance the quality of data obtained with the system described in Chapters II and III. First of all, our camera calibration technique, while sufficient, is cumbersome in that it relies on photographing a target submersed in syrup that is manually and precisely moved in known fashion. Methods have been developed for the self-calibration of cameras [5, 6] that would potentially eliminate this tedious dependence on a target and permit calibration to be performed on the fly. Since measuring and controlling the location of a target is not required, self-calibration also

has the potential to further reduce PIV errors that arise due to uncertainty between the true and measured target position.

As for the remaining avenue of improvement, a major criterion in the design of our PIV system and data processing software has been robustness to experimental deficiencies. While there is no substitute to a well-executed experiment, unforeseen circumstances can always occur that degrade data. In some cases, we can replace a degraded dataset by simply running an experiment again. In many other cases, recollecting data is not practical, and even when it is, there is no guarantee that new problems will not be encountered. Consequently, a degree of tolerance built into the experimental systems helps ensure that all data are leveraged to the fullest extent possible. These considerations were the primary motivation behind development of the hybrid PIV/PTV scheme discussed in Chapter III. A number of proposals for augmenting the hybrid scheme were presented in the chapter, and any of those modifications would greatly extend the capabilities of the technique.

Additional experiments

During our investigations, several aspects of plume dynamics were identified as warranting additional study. Overall, the full implications of the nonlinearities introduced by temperature dependent viscosity on the dynamics of thermal plumes remains poorly understood. As mentioned in Chapter IV, the decrease in strain rate normal to the plume leading edge manifold as a function of time for the warmest plumes studied seems conceptually linked to the temperature dependent viscosity of the syrup. From the standpoint of stretching efficiency, the falloff in strain rate suggests that a hot starting plume is more destructive earlier in its life than later, and that a cold starting plume's destructive capacity remains unchanged throughout its evolution. However, this is but one example of phenomena that may be tied to temperature dependent rheology. As a very vigorous plume evolves from inception, to liftoff, to impacting

the upper surface, it passes through a number of different flow regimes. In some of these regimes, dynamics that are at work in a real flow may have pronounced effects that are necessarily neglected in our theoretical or numerical investigations of plume behavior. Additional experimental datasets that explore different viscosity contrasts would greatly enhance our understanding of the importance temperature dependent rheology plays on mass transport, plume formation time scales, and stirring.

It is also possible that not all plumes generated in Earth's interior have sufficient initial buoyancy to survive [3]. One of our most intriguing datasets was of a failed plume (ref. Chapter II), but the precise mechanisms that lead to the plume collapse in our particular case are not well-understood. What is clear, however, is that such events, if they occur, have the potential to significantly redistribute heterogeneity in the mantle. A large, relatively cold upwelling that traverses only part of the mantle has the potential to deplete a fairly significant source reservoir and transport that material to a zone within the surrounding mantle that may not serve as source for a plume or otherwise be sampled and expressed at the planetary surface for potentially long times. Further exploration of the conditions necessary to produce failed plumes as well as the mass transport properties of such events would be very beneficial.

That Earth's liquid outer core provides a free slip boundary condition is also of great interest. Our experimental apparatus realized a no-slip condition on all surfaces, and such boundary conditions might rightfully be expected to have some impact on mass transport within the viscous boundary layer near the heater. Although not discussed in the current document, our analyses did indeed suggest a very distinct and different relative orientation of mass within our plumes compared to that seen by Farnetani *et al.* [2] using a numerical model of thermal plumes generated with a free-slip lower boundary. In our experiments, the fluid which wetted the tank bottom was the least likely to become incorporated into the mature plume. On the other hand, fluid adjacent to the warm, free-slip lower boundary of Farnetani *et al.*

[2] is preferentially drawn into the upwelling since it has the lowest viscosity (*i.e.*, it is the hottest mantle fluid) and does not need to work against a no-slip surface. The presence of a free-slip lower boundary condition may also have implications for the shape of the Lagrangian coherent structures (ref. Chapter IV) observed in our experiments. Although experiments exploring a free-slip lower boundary would be expensive, the possibility of such work has been explored by the present author and certainly seems feasible.

The presence of a well-defined forward-time LCS surrounding the plume generated via injection of hot syrup (ref. Chapter IV) and the lack of such a structure in our plumes generated via localized heating is extremely interesting. As discussed in Chapter IV, the injection plume's outermost LCS has profound implications for the manner in which the starting plume head entrains mass. It is not, however, abundantly clear from our work that thermal plumes generated via localized heat sources are prohibited from possessing such structures. There may be operating regimes whereby plumes are generated with high Rayleigh numbers through impulsive heating using a localized heat source that result in the presence of a forward-time LCS which surrounds the bottom of the plume head. Discovering the precise conditions necessary to produce such a forward-time LCS would be an immensely powerful tool for the classification of thermal plumes and greatly assist the design of future experiments for studying the dynamics of such flows.

Concluding remarks

Even simple flows like the laminar plumes studied in this thesis show a surprising degree of complexity, and such observations serve as a constant reminder of the difficulty scientists face in trying to untangle the mysteries of planetary evolution. The challenge is made all the more arduous by the remoteness of planetary interiors, but persistence will undoubtedly lead to new and exciting discoveries.

The work we have conducted analyzing the mass transport properties of thermal plumes is a small step toward a better understanding of how the material from which Earth accreted has been transformed by time. At least over the parameter space explored, the heads of axisymmetric starting plumes generated by a localized heat source are ineffective at entraining overlying mass. Furthermore, the plume heads do not exhibit any pronounced ability to mechanically stir mass within the head. The leading edge of the starting plume, however, has the potential to be quite destructive from a molecular mixing perspective due to the strong stretching which occurs.

As discussed in Chapters IV and V, sidewall effects have impacted our results on many levels which makes developing or extending quantitative theories a challenge. It is, for example, quite difficult to determine whether the velocity scaling of Eq. 4.26 adequately captures the effects of temperature-dependent viscosity when sidewalls reduce the expected rise velocity of the plume by up to 50% (ref. Chapter IV). The presence of sidewalls has no appreciable impact on the above qualitative observations regarding how the plumes entrain mass, but the statistics of precisely how much mass comes from a certain location would change. The seemingly large sidewall effects are an unfortunate aspect of studying high Prandtl number fluids in the laboratory, but is important to keep in mind that uncertainty in many mantle parameters is an order of magnitude or more [4]. Given our present state of knowledge, no laboratory experiment (or highly sophisticated numerical model) can provide an exact representation of mantle flows. But even if we cannot precisely compute where the mass of flood basalts can originate in the mantle based on the assumption that such surface features are produced by plume heads, we still gain much insight by knowing the plume head composition is not dominated by mass located above the rising plume (ref. Chapter IV).

Regarding experimental technique, much effort has been expended to develop our hardware and software such that the message provided by the data are impacted as

little as possible by our preconceived notions of how we expect the data to look. These considerations are the primary motivation for developing the hybrid PIV/PTV scheme (ref. Chapter III) and using the displacement field to iteratively deform the image of flow regions having high velocity gradients (ref. ChapterII). Instead of simply removing spurious vectors with a filter, we have opted to employ these more computationally expensive techniques to leverage the information contained in the data.

References

[1] A. Davaille, A. Limare, F. Touitou, I. Kumagai, and J. Vatteville. Anatomy of a laminar starting thermal plume at high Prandtl number. *Experiments in Fluids*, pages 1–16, 2010.

[2] C. G. Farnetani, B. Legras, and P. J. Tackley. Mixing and deformations in mantle plumes. *Earth and Planetary Science Letters*, 196:1–15, 2002.

[3] I. Kumagai, A. Davaille, K. Kurita, and E. Stutzmann. Mantle plumes: Thin, fat, successful, or failing? constraints to explain hot spot volcanism through time and space. *Geophysical Research Letters*, 35(16):L16301, 2008.

[4] G. Schubert, D. L. Turcotte, and P. Olson. *Mantle Convection in the Earth and Planets*. Cambridge University Press, 2001.

[5] B. Wieneke. Volume self-calibration for 3D particle image velocimetry. *Experiments in Fluids*, 45(4):549–556, 2008.

[6] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

APPENDICES

APPENDIX A

Dynamical systems

A.1 Introduction

The dynamical systems literature is very large, being spanned by works that are devoted entirely to the development of the underlying mathematical foundations to numerous papers and a few books that are truly focused on applications of the theoretical concepts to real-world problems. This appendix should not be viewed in any way as providing complete, rigorous instruction on dynamical systems theory. The rather simple goal of the present document is to provide a short, somewhat self-contained introduction to those elements of dynamical systems theory that the author found most useful when navigating the applied literature. For a more general and thorough perspective of dynamical systems, there are several very useful books available (*e.g.*, [1, 35, 17, 39, 13, 27]).

The current interest in dynamical systems theory traces its roots back to the very work of the field's principal founding father, Poincaré. In the late 1800's, Poincaré was investigating a problem that had been troubling some of the greatest scientific minds for centuries: the stability of the solar system. The difficulty of the problem derived directly from the nature of the governing differential equations — gravity was found to behave as an inverse square law, and hence the differential equations of motion were nonlinear. Then as now, finding analytical solutions to nonlinear

differential equations is a challenging task.

Up until the time of Poincaré, most theoretical work on celestial mechanics had centered around attempts to find explicit expressions for positions of the orbiting bodies as a function of time. Investigators' optimism was no doubt buoyed by the successes encountered with the analysis of the much simpler two body problem. Indeed, the equations of motion for two bodies interacting via gravity had been solved by Newton himself [3]. Later work on the topic would actually permit Bertrand to show that perturbed orbits remain closed (*i.e.*, the motion is periodic[1]) in a two-body power law central force problem if and only if the central force has one of two forms [12]: Hooke's law where force varies in linear proportion to the distance between the bodies, or an inverse square law where force is inversely proportional to the square of the distance between the bodies. Unfortunately augmenting the two body planetary model with a single additional gravitationally interacting body, let alone a full solar system packed with planets and planetary moons, makes solving the three body system so difficult that it has to date eluded a general, closed-form solution.[2]

Recognizing the problems countless others had faced before him, Poincaré decided to investigate the three-body system from a different perspective. Instead of attempting to find explicit solutions for the positions of the three bodies as a function of time given some initial conditions, Poincaré queried the qualitative nature of all solutions using geometric arguments and the asymptotic behavior of trajectories in the infinite time limit. By trading off the ability to say definitively where planet a would be at time t, Poincaré's tools and those developed later by others provided the capacity to make qualitative, but nonetheless significant, characterizations about the nature

[1] An unclosed orbit does not imply that the orbit is unbounded and flies off to infinity, but merely that the motion of the orbiting body is not periodic (*i.e.*, the orbit never retraces its path).

[2] An infinite series solution to the three-body problem was determined by Sundman [36], however such infinite series expansions are typically not considered closed-form solutions. Moreover, Sundman's expansion is so slowly converging that it provides essentially no insight into the behavior of the orbits. Approximately $10^{8,000,000}$ terms of Sundman's series would be required to compute a three-body orbit with the accuracy typical of present day ephemeris measurements [38].

of an orbit given the system's initial conditions. Are there certain configurations of the three bodies that always result in at least one flying off to infinity? What initial configuration is necessary for a third body to be captured by one of the other two? Are there regions in space that the bodies are forever prohibited from occupying? Dynamical systems theory provides the framework for resolution of these questions.

Although the study of fluid mechanics is in many ways vastly more complex than orbital mechanics, the two fields share one very important trait: the governing equations are nonlinear. Furthermore, fluid dynamicists face many of the same or similar questions encountered in orbital dynamics. Are there regions in a given flow that have distinct behavior? If so, how can one identify them, and how much if any mass is exchanged between these distinctive zones? Are there regions of the flow field that are completely isolated from stirring with the ambient? How does a fluid parcel transiting a certain area behave? Does the parcel eventually loop back onto its starting point, or does the parcel perhaps wind around through time eventually coming close to every spatial location in the flow domain under analysis while never quite retracing its previous path?

To provide a concrete example of how some of these questions can arise in a fluid mechanics context, consider the velocity field of the periodically forced Duffing equations

$$\begin{aligned} u_x &= y \\ u_y &= x - x^3 + \epsilon \cos \omega t, \end{aligned} \tag{A.1}$$

where ϵ is a parameter that controls the magnitude of the forcing, and ω is the angular velocity of the forcing function. The velocity field given by Eq. A.1 satisfies $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$ and can be thought of as an incompressible 2D flow arising from some appropriate driving conditions. Following [30], we set $\epsilon = 0.5$ and $\omega = 3$ yielding a forcing period

of $T = 2\pi/3$ time units. The corresponding velocity field is shown in Figure A.1. Assume that the flow has existed and will exist for all time, and further suppose that at $t = 0$ two blocks of fluid are instantaneously dyed different colors and released into the flow. Figure A.2 presents three different scenarios. In each case, dyeing of the fluid has been simulated by filling the different regions with a large number of blue or red passive tracers. These individual passive tracers are then advected with the flow using a fourth order Runge-Kutta algorithm for 16 periods of the forcing function. What happens to the blocks and how they interact is not immediately obvious from examining the velocity field alone.

The blocks of Figure A.2(a) intermingle and become well-stirred.[3] But why? Perhaps the observed behavior is simply a result of the initial proximity of the colored blocks. The blocks, after all, remain isolated if the initial separation is greater, as in A.2(c). Unfortunately if the block separation along the x-axis is increased once more, efficient stirring again ensues as captured in Figures A.2(e,f). So clearly initial proximity of the tracer blocks cannot be a dominating factor affecting stirring within this particular flow after all. With further reflection, we may conclude that the scenario of Figure A.2(c) keeps the blocks isolated because they are released into part of the velocity field that seems to recirculate (ref. Figures A.1(b,d)). The other two scenarios perhaps stir well solely because the initial location of the blocks lie outside these recirculation zones. Indeed through close examination we can see that the pattern of stirred fluid shown in Figure A.2(b) is almost identical to that of Figure A.2(f). We therefore might posit that this flow will efficiently stir all fluid as long as the material to be stirred is located sufficiently far away from the recirculation zones. However as demonstrated by Figure A.3, our hypothesis is terribly wrong. Although the evolved tracer field of Figures A.3(b-d) is only shown at three time values following

[3]Here and throughout, the notion of stirring will be used to represent the diffusionless process of stretching and folding fluid parcels. Mixing, on the other hand, will be reserved for describing processes where molecular diffusion is important.

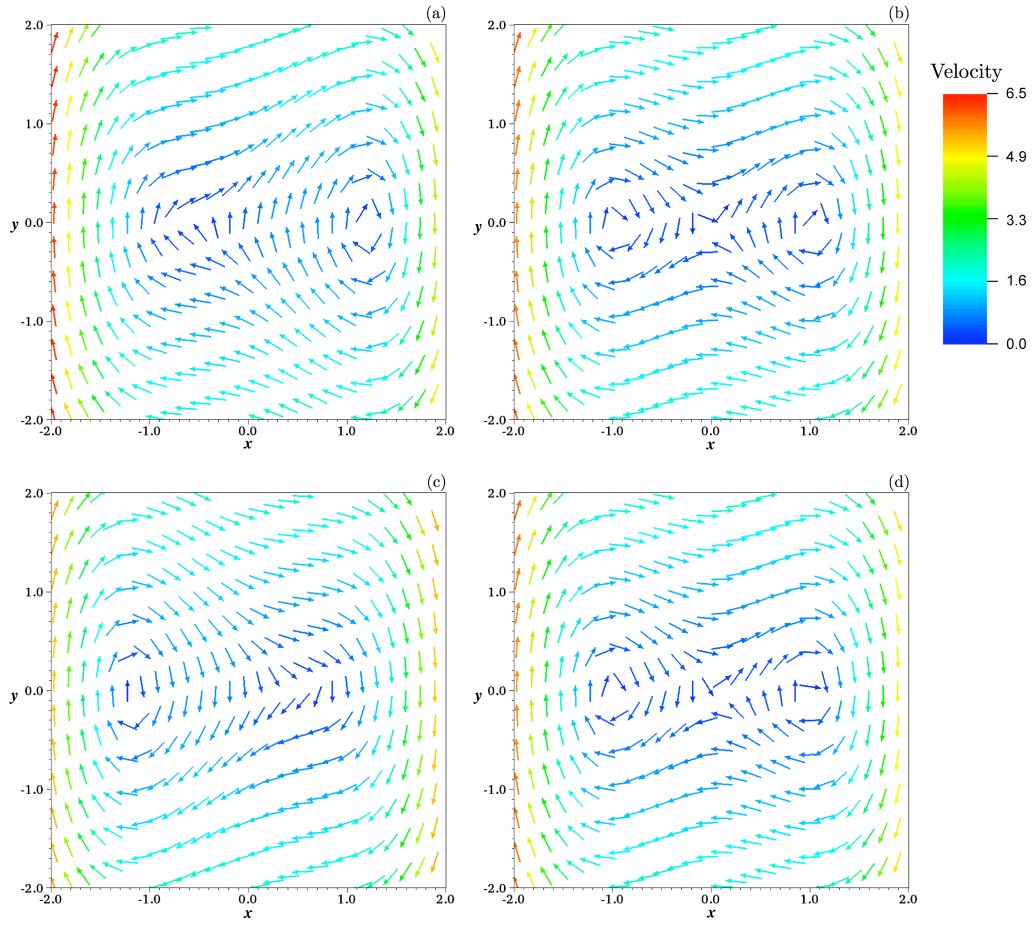

Figure A.1: Velocity field of the periodically forced Duffing equations at four time values: (a) $t = 0$, (b) $t = \pi/6$, (c) $t = \pi/3$, (d) $t = \pi/2$. For clarity, the velocity vectors are of equal length and colored by magnitude.

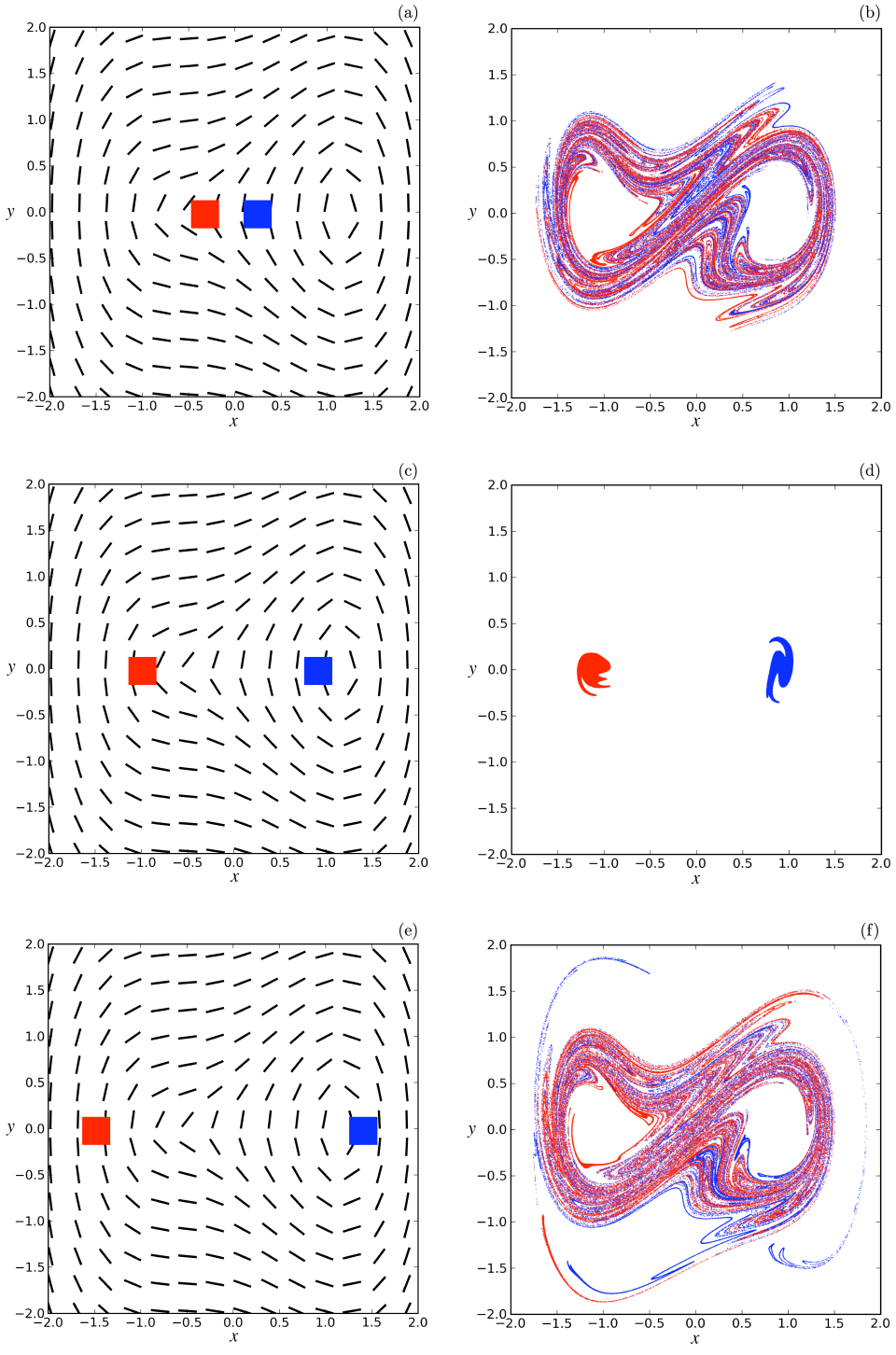

Figure A.2: Three scenarios, one for each row of the figure, demonstrating the evolution of two groups of passive tracers (red and blue) under the periodically forced Duffing equations. (a, c, e) Initial distribution of tracers at $t = 0$ for each scenario with black line segments drawn tangent to the instantaneous velocity field for reference. (b, d, f) Corresponding tracer fields after an elapsed time equivalent to sixteen forcing periods ($t = 32\pi/3$).

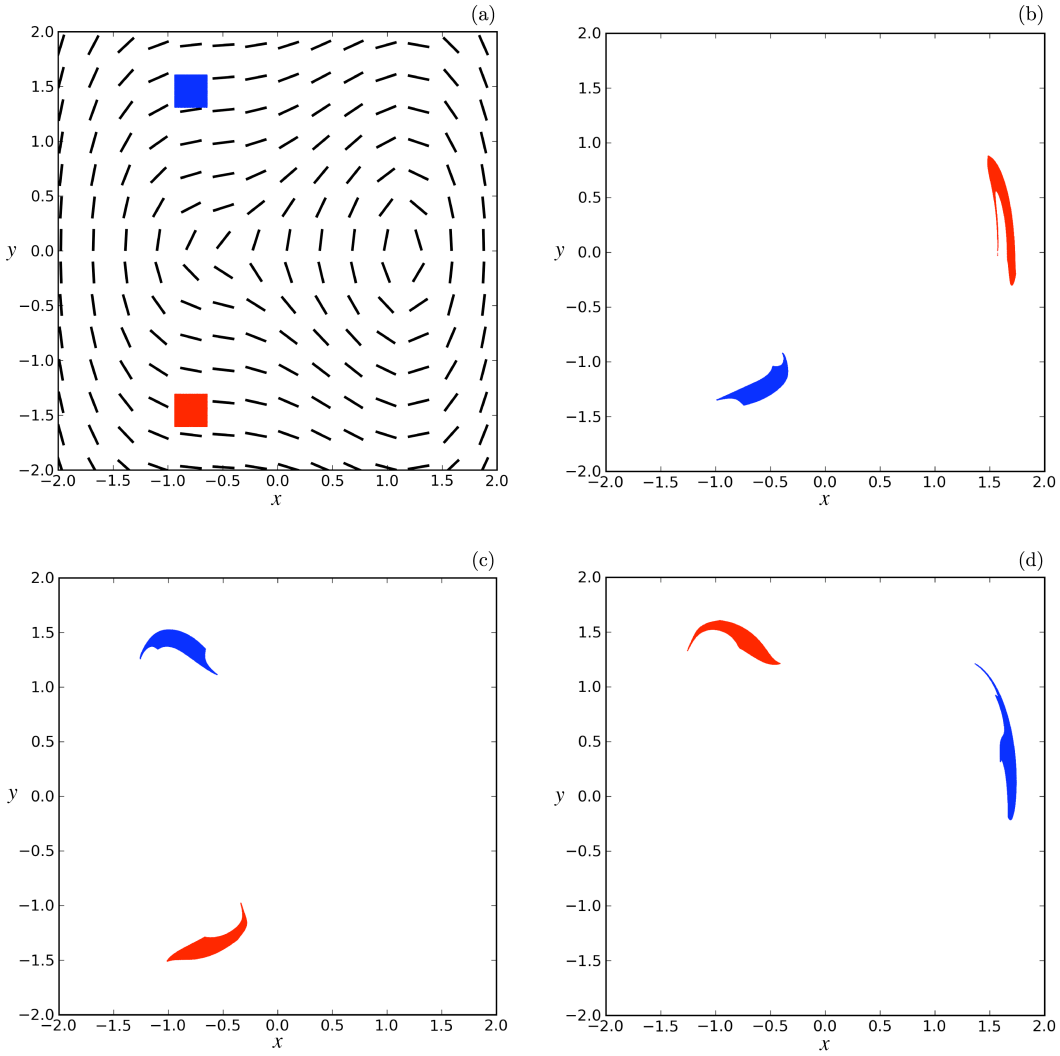

Figure A.3: Evolution of two groups of passive tracers (red and blue) under the periodically forced Duffing equations. (a) Initial distribution of tracers at $t = 0$ with black line segments drawn tangent to the instantaneous velocity field for reference. (b, c, d) Tracer fields after fourteen ($t = 28\pi/3$), fifteen ($t = 10\pi$), and sixteen ($t = 32\pi/3$) forcing periods, respectively.

initial tracer release, the behavior illustrated persists for all time: the red and blue blocks rotate clockwise around the flow domain without ever interacting with each other. Apparently the example flow field supports mysterious regions far away from the areas of recirculation where stirring is also nonexistent.

As an additional motivating example, consider the flow of a periodically perturbed Hill's spherical vortex, where the perturbation can be imagined as perhaps arising from a wavy walled pipe through which the spherical vortex is propagating. Let the vortex initially propagate along the z-axis and then change to a coordinate system that moves with the mean velocity of the vortex. The equations for the resulting perturbed model velocity field are [6]:

$$\begin{aligned} u_x &= (u_r \sin\theta + u_\theta \cos\theta)\cos\phi - \frac{\epsilon}{2}\sin(\omega t) \\ u_y &= (u_r \sin\theta + u_\theta \cos\theta)\sin\phi - \frac{\epsilon}{2}\sin(\omega t) \\ u_z &= u_r \cos\theta - u_\theta \sin\theta + \epsilon \sin(\omega t), \end{aligned} \tag{A.2}$$

with

$$u_r = \begin{cases} -\frac{3}{2}u_0\left(1 - \frac{r^2}{a^2}\right)\cos\theta, & r < a \\ u_0\left(1 - \frac{a^3}{r^3}\right)\cos\theta, & r \geq a \end{cases} \tag{A.3}$$

$$u_\theta = \begin{cases} \frac{3}{2}u_0\left(1 - 2\frac{r^2}{a^2}\right)\sin\theta, & r < a \\ -u_0\left(1 + \frac{a^3}{2r^3}\right)\sin\theta, & r \geq a. \end{cases} \tag{A.4}$$

Note that a is the vortex radius, u_0 the free stream velocity, $r = \sqrt{x^2 + y^2 + z^2}$, $\theta = \arccos(z/r)$, $\phi = \arccos(x/\rho)$, and $\rho = \sqrt{x^2 + y^2}$. The parameter ϵ controls the magnitude of the perturbation and ω its frequency. As with the Duffing oscillator, the perturbed Hill's spherical vortex satisfies $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$. A 2D cross section of the velocity field at the $y = 0$ plane for $a = 1$, $u_0 = -1.5$, $\epsilon = 0.3$, and $\omega = 2\pi$ is provided in Figure A.4.

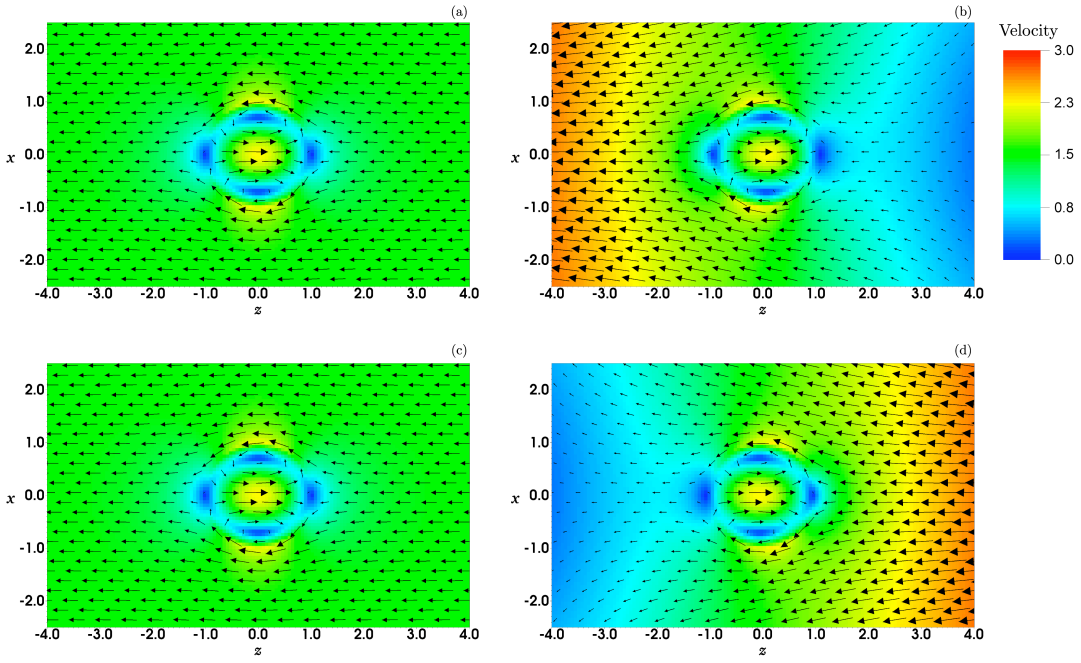

Figure A.4: Velocity field for the periodically forced Hill's spherical vortex at four time values: (a) $t = 0$, (b) $t = 1/4$, (c) $t = 1/2$, (d) $t = 3/4$. Velocity magnitude is represented by the length of the vectors as well as the pseudocolor background.

Releasing the multi-colored blocks of fluid depicted in Figure A.5(a) into this velocity field at $t = 0$ and allowing the flow to evolve for eight forcing periods produces what at first glance may appear to be very strange behavior as illustrated in Figures A.5(b,c). While the orange and cyan sections pass over the vortex quickly, the red and blue regions get entrained into the vortex where they are repeatedly stretched and folded while wrapping around but never penetrating the remaining vortex core. Also of interest is the distinct lobe shape the red and blue fluid takes, which somewhat resembles the tendrils of a jellyfish, as it exits the vortex. Perhaps the most peculiar observation is that the orange and red fluid, which initially touch, do not subsequently intermingle to any significant degree. On the contrary, the red and blue fluids are initially separated by a comparatively large distance but stir quite effectively. Can we explain the stirring behavior of Hill's spherical vortex and provide a robust means of predicting what material participates?

In the following sections we will develop a few critical dynamical systems concepts and return to the example flows of the Duffing oscillator and Hill's spherical vortex. As we shall see, dynamical systems theory provides the keys to discerning why particular regions of a flow field stir well and why some zones remain isolated for all time. We will even discover why the stirred fluid in the above examples takes the shape observed in the provided figures. But first, we must solidify some terminology.

A.2 The language of dynamical systems theory

Dynamical systems theory, as with most fields of study, has its own vocabulary. While most of the needed language will be developed in turn, there are a few core items that need to be established before proceeding further.

Obviously dynamical systems theory is concerned with systems, but what exactly do we mean by the term dynamical system? A *dynamical system* is generally taken to be the combination of a state space and a rule that governs how the state of the

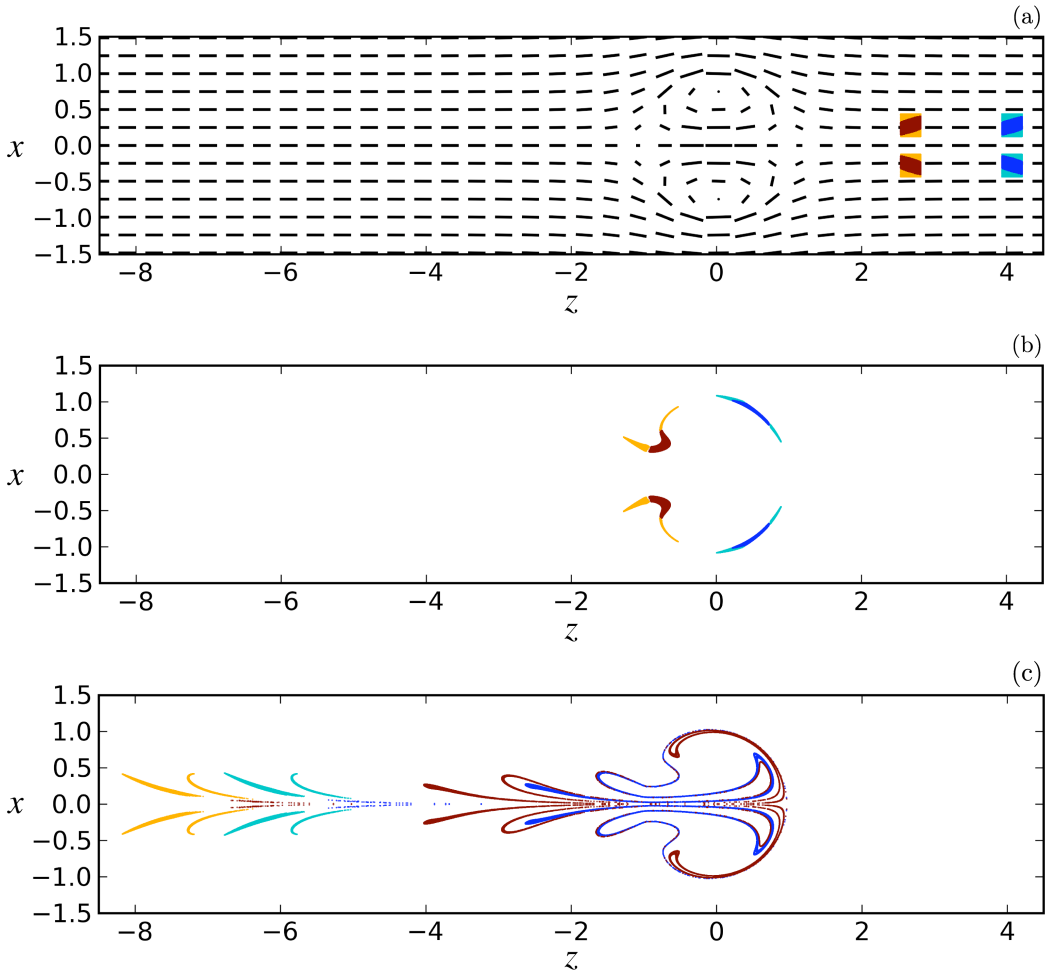

Figure A.5: Evolution of colored passive tracer blocks subjected to the periodically forced Hill's spherical vortex velocity field. (a) Initial tracer distribution at $t = 0$ with line segments (black) proportional in length and tangent to the instantaneous velocity field provided for reference. (b, c) Tracer distribution after three ($t = 3$) and eight periods ($t = 8$) of the forcing, respectively.

system evolves given some initial conditions [1]. *State space* itself consists of all states a system can occupy and is spanned by the minimum set of variables necessary to define the state of the system sufficiently such that its future state can be predicted provided the current state is known.[4] For a fluid flow of interest, we might choose as our dynamical system the combination of the Navier-Stokes equations and a state space spanned by temperature, velocity, density, and possibly time if the flow is time-dependent. Or, if we are particularly interested in mass transport, we could instead consider tracking the spatial position of passive tracers evolving in a specified velocity field, as was done in Section A.1. In this case, the state space is spanned by spatial coordinates of the tracers (and potentially time if the velocity field is unsteady). The rule governing the change in tracer position becomes

$$\dot{\boldsymbol{x}} = \frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t), \tag{A.5}$$

where $\boldsymbol{x}$ is the particle position, and $\boldsymbol{f}(\boldsymbol{x}, t)$ is the fluid velocity at that position and some time, t.

In regard to the rule governing system evolution, two dominant types are encountered in the literature: iterated maps and differential equations. Generally speaking, iterated maps function in discrete time while differential equations operate in continuous time. While that may be a subtle distinction, the consequences can be very important. Discrete time systems with a given state space dimension are able to support more complex behavior than their continuous time cousins of equal dimension [17]. The reason for the difference in behavior is a consequence of the map trajectory's ability to jump around in state space without violating the no-intersection theorem, whereas a solution from a continuous-time differential equation is prohibited from ever crossing itself.[5] This appendix focuses exclusively on systems described by dif-

[4]Some authors refer to state space as phase space, while others reserve the term phase space solely for Hamiltonian systems. The two are equivalent for our purposes.

[5]To do so would create a single state of the system at the point of intersection that simultaneously

ferential equations, nevertheless, iterated maps are important dynamical systems in their own right and can also be quite useful in the study of periodic, continuous time systems by way of Poincaré sections.

Finally, dynamical systems theory provides several prominent names for a solution to the set of differential equations governing system evolution. We will refer to the solution of a system of equations as a *trajectory* through state space, although *orbit* is also very common in the literature.

A.3 Fixed points

With the basic terminology available we can now start the dynamical systems development. Let us begin with the system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, t). \tag{A.6}$$

Eq. A.6 constitutes a system of first order ordinary differential equations with the right hand side (RHS) being a function of the dependent variable, $\boldsymbol{x}$, and the independent variable, t. When the independent variable, in this case time, shows up explicitly in the RHS, the system is classified as *non-autonomous*. If the RHS is independent of time, the system is deemed *autonomous* and can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}). \tag{A.7}$$

Note that non-autonomous systems can be made autonomous by simply introducing a new state space variable for t. Equations A.1 and A.2 from the Introduction, for example, are both non-autonomous systems. They can be made autonomous by introducing a new state space variable, $\theta = \omega t$, and augmenting the system of

produces two different outcomes – a violation of the uniqueness of solution theorem [1, Thrm. 7.14].

differential equations with $\dot{\theta} = \omega$. In a sense, then, autonomous systems are the more general form.

The number of equations used to describe the system is equal to the dimension of the state space, with each differential equation specifying how the underlying state space variable changes with time. Any group of differential equations, be they higher order ordinary differential equations or partial differential equations, can in principle be broken down into a set of first order ordinary differential equations. In the case of partial differential equations, the resulting set of first order ODE's may be infinitely large, but nevertheless the transformation is conceptually possible. The rationale for why a partial differential equation results in an infinite dimensional state space should be intuitively clear from the definition of a state space given in Section A.2. Initial and boundary conditions for partial differential equations are specified using continuous functions as opposed to discrete values, and given that a continuous function can be evaluated at infinitely many points, we therefore require infinitely many pieces of information to fully specify the state of a system governed by partial differential equations.

We will only make one requirement of our generic system given by Eq. A.7 — that $\boldsymbol{f}$ be Lipschitz continuous. On some open subset U of $\mathbb{R}^n$, the function $\boldsymbol{f}$ is Lipschitz continuous provided a constant K exists such that

$$|\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}_2)| \leq K\,|\boldsymbol{x}_1 - \boldsymbol{x}_2| \tag{A.8}$$

is valid for all $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in U. The constant K is known as the *Lipschitz constant* and as demonstrated in Figure A.6 can be thought of in the 1D case as a sort of limiting slope that constrains how rapidly $\boldsymbol{f}$ can vary as a function of $\boldsymbol{x}$. If $\boldsymbol{f}$ is Lipschitz on U, then by the existence and uniqueness theorem of ordinary differential equations, a solution exists, and it is unique [18, Ch. 17.2]. Furthermore, if $\boldsymbol{f}$ is Lipschitz on U

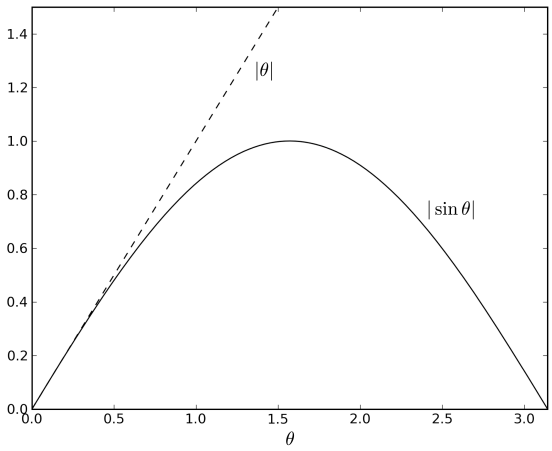

Figure A.6: Lipschitz continuity in 1D for $f(\theta) = \sin\theta$. Note that for all θ in $\mathbb{R}$, $|f(\theta)| \leq |\theta|$. Hence $\sin\theta$ is Lipschitz continuous with $K = 1$.

with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ representing two separate solutions of Eq. A.7 that are valid on the time interval $[t_0,\, t_1]$, then

$$|\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t)| \leq |\, \boldsymbol{x}_1(t_0) - \boldsymbol{x}_2(t_0)|\, e^{K(t-t_0)} \tag{A.9}$$

for all t in $[t_0,\, t_1]$. Eq. A.9 is a statement of the theorem regarding continuous dependence of the solution on initial conditions [18, Ch. 17.3].

The existence and uniqueness theorem places a very significant constraint on trajectories through state space: solutions must be unique, and hence trajectories can never intersect. The importance of Eq. A.9, on the other hand, is the limit it places on the rate of separation growth or decay for nearby trajectories. Namely, while two solutions may certainly move toward or away from each other as time evolves, they may do so at most exponentially fast. This notion of the distance between nearby trajectories varying at most exponentially will be seen again later in the discussion of Lyapunov exponents.

There may be certain critical values of $\boldsymbol{x}$ for which $\boldsymbol{f}(\boldsymbol{x}) = 0$. From Eq. A.7, we then see that $\dot{\boldsymbol{x}} = 0$ at these critical points. Consequently, the critical values are

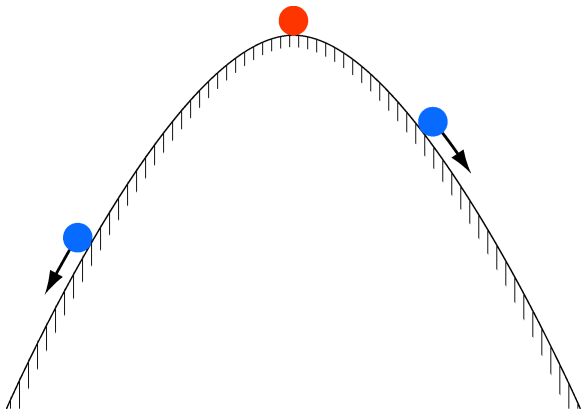

Figure A.7: The blue ball can be placed almost anywhere on the hill and it will subsequently roll down it. The very apex of the mound, however, is an equilibrium point, and a ball placed there will remain there for all time provided the ball is never perturbed.

equilibrium points, and a system initialized on a critical point will remain there for all time. Given the stationary character of a system initialized at one these critical values, they are widely referred to as *fixed points* of the system.

As pointed out in Section A.1, a goal of dynamical systems theory is the ability to describe the behavior of all trajectories in some region without actually solving the system given by Eq. A.7. Determining how trajectories behave in the vicinity of a fixed point is one example, albeit an extremely important one, of how dynamical systems concepts can provide significant insight into general system behavior. It should come as no surprise that equilibrium points, if they exist, have significant influence on the overall behavior of a system. Think of a ball perched on top of a perfectly smooth, isolated mountain and acting under the influence of gravity (Figure A.7). Any perturbation will clearly cause the ball to roll down the hill, but the mere presence of the equilibrium, and hence the corresponding hill, also determines in a generic sense how a ball placed somewhere else on the hill, potentially far from the equilibrium, behaves. Namely, the ball will move away from the equilibrium by rolling down the hill.

In order to determine whether trajectories move toward, away from, or orbit a

particular fixed point in an arbitrary system, we need to characterize the stability of the fixed point. We do this by investigating the behavior of trajectories in the near vicinity of the fixed point in question. Consider Eq. A.7 for an n-dimensional system and let $\boldsymbol{x}^*$ represent a fixed point of the system. We want to analyze the motion of another trajectory, $\boldsymbol{x}_b$ near the fixed point. Provided that $\boldsymbol{x}_b$ is sufficiently close to $\boldsymbol{x}^*$, we can construct a set of linearized differential equations that in many cases will permit successful classification of the fixed point's stability. Clearly $\boldsymbol{x}_b$ is a solution to the autonomous system, so

$$\dot{\boldsymbol{x}}_b = \boldsymbol{f}(\boldsymbol{x}_b). \tag{A.10}$$

Linearizing about $\boldsymbol{x}^*$, we find

$$\dot{\boldsymbol{x}}_b = \boldsymbol{f}(\boldsymbol{x}_b) = \boldsymbol{f}(\boldsymbol{x}^*) + \boldsymbol{\nabla f}|_{\boldsymbol{x}^*}\, \delta\boldsymbol{x} + O(|\delta\boldsymbol{x}|^2) \tag{A.11}$$

where $\delta\boldsymbol{x} = \boldsymbol{x}_b - \boldsymbol{x}^*$. Rearranging Eq. A.11 and dropping the higher order terms yields

$$\delta\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{x}^*)\delta\boldsymbol{x} \tag{A.12}$$

with $\boldsymbol{J}(\boldsymbol{x}^*) = \boldsymbol{\nabla f}|_{\boldsymbol{x}^*}$. Given that $\boldsymbol{x}^*$ is a fixed point (*i.e.*, it is a constant), $\boldsymbol{J}(\boldsymbol{x}^*)$ is a matrix of constant coefficients. Consequently a general solution to the linearized system of Eq. A.12 is readily found and of the form

$$\delta\boldsymbol{x}(t) = c_1\boldsymbol{\eta}_1 + c_2\boldsymbol{\eta}_2 + \ldots + c_n\boldsymbol{\eta}_n \tag{A.13}$$

where the $\boldsymbol{\eta}_i$ are n, linearly independent solutions that depend exponentially on the eigenvalues of $\boldsymbol{J}$ but may also be combined with polynomials in t. That is

$$\boldsymbol{\eta}_i(t) = \boldsymbol{g}_i(t)e^{\lambda_i t}, \quad i = 1, \ldots, n \tag{A.14}$$

where λ_i is an eigenvalue of $\boldsymbol{J}$, and $\boldsymbol{g}_i(t)$ may be a polynomial of time.[6] For large t, the exponential term of Eq. A.14 dominates any polynomial, and we can immediately see that the eigenvalues of $\boldsymbol{J}$ control the dynamics of the linearized system.

Therefore if all eigenvalues of $\boldsymbol{J}$ at $\boldsymbol{x}^*$ are real and negative, then the general solution (Eq. A.13) of the linearized system will go to zero as $t \to \infty$. The fixed point about which the system was linearized is subsequently classified as a *stable node*. Conceptually, we can perturb the autonomous system away from a stable node, and its state will asymptotically evolve back to that of the node. In contrast, if all eigenvalues of the linearized system are positive and real, small displacements $\delta\boldsymbol{x}$ from the fixed point will grow with time. Here the fixed point of the autonomous system is classified as an *unstable node*. If all eigenvalues are real, nonzero, and of differing signs, then nearby trajectories will approach the *saddle* fixed point in some directions and recede from it along others. Complex eigenvalues identify fixed points that form spirals or centers. If the complex eigenvalues have nonzero real parts, then nearby trajectories will recede from or decay to a *spiral node* depending on whether the real parts are positive or negative, respectively. Finally, a fixed point with purely imaginary eigenvalues is referred to as a *center,* and trajectories in a neighborhood of the point will form closed orbits around it.[7] Figure A.8 presents several examples of different fixed point types in 2D state space.

One final note regarding fixed point classification is in order before moving on to an example. Collectively, those fixed points where all eigenvalues of $\boldsymbol{J}$ have non-zero real parts are referred to as *hyperbolic* points. In contrast, the eigenvalues of *elliptic* points are purely imaginary.[8]

[6] See a text on ordinary differential equations (*e.g.*, [5]) for more.

[7] If the real part of any eigenvalue is zero, then additional techniques are generally required to robustly classify the fixed point. In the case of a nonlinear system, the nonlinear terms that have been neglected during the linearization can alter the true character of fixed points having eigenvalues with $\mathrm{Re}(\lambda) = 0$. These nonlinear terms can easily transform what appears to be a center, for example, of the linearized system into a spiral node where trajectories spiral toward or away from the fixed point of the true nonlinear system.

[8] Although we have yet to discuss Lyapunov exponents, they can be used to extend the notion of

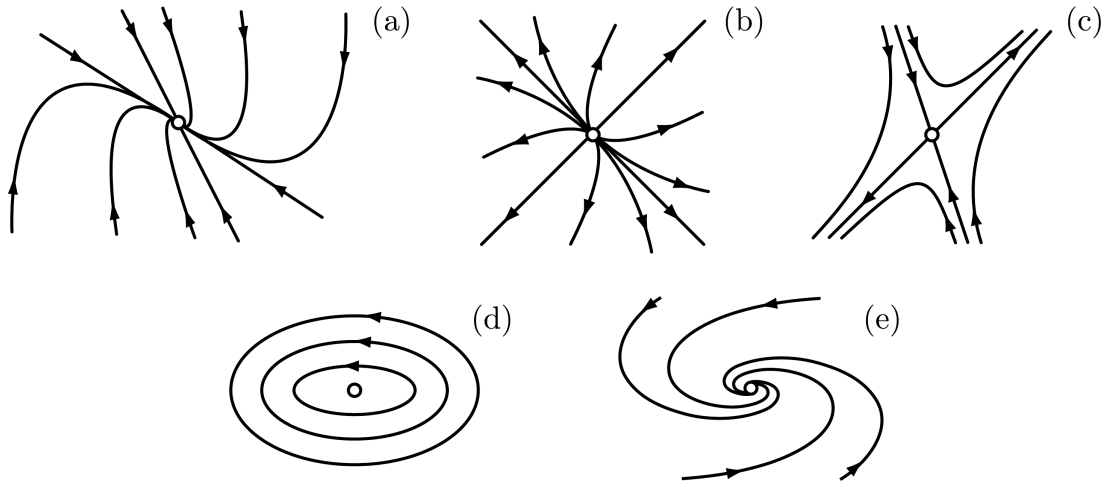

Figure A.8: *Phase portraits* of several example fixed points in a 2D state space. For each case, the fixed point is identified by the open circle, and several trajectories are illustrated with arrows depicting the direction of trajectory evolution. (a) Stable node. (b) Unstable node. (c) Saddle point. (d) Center. (e) Spiral node.

Now for an example. Consider the unforced Duffing system of Eq. A.1 with $\epsilon = 0$. We then have the system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} y \\ x - x^3 \end{pmatrix}, \tag{A.15}$$

with three fixed points at $\boldsymbol{x}^* = (0,0), (\pm 1, 0)$. The gradient of $\boldsymbol{f}$ becomes

$$\boldsymbol{J} = \boldsymbol{\nabla} \boldsymbol{f} = \begin{pmatrix} 0 & 1 \\ 1 - 3x^2 & 0 \end{pmatrix} \tag{A.16}$$

with eigenvalues, λ, of $\boldsymbol{J}$ satisfying $\lambda^2 + 3x^2 - 1 = 0$. Consequently, the fixed point at the origin is a hyperbolic saddle point with eigenvalues $\lambda_{1,2} = \pm 1$ and the real

hyperbolicity to linearizations where $\boldsymbol{J}$ is a function of time. Such scenarios arise when determining if arbitrary trajectories or other state space structures attract or repel nearby trajectories. See Section A.4 and [40] for more details.

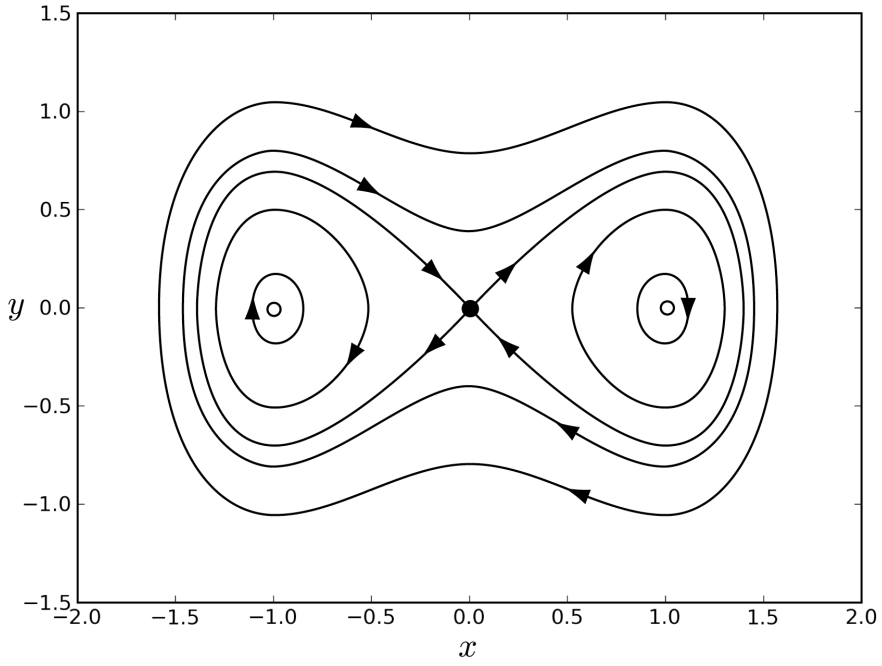

Figure A.9: Phase portrait for the unforced Duffing oscillator. The elliptic points at $\boldsymbol{x}^* = (\pm 1, 0)$ are indicated by small open circles, while the hyperbolic point at the origin is shown as a solid circle.

eigenvectors

$$\boldsymbol{v}_1 = \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix}, \quad \boldsymbol{v}_2 = \begin{pmatrix} -1 \\ \\ 1 \end{pmatrix} . \tag{A.17}$$

In contrast, the fixed points at $\boldsymbol{x}^* = (\pm 1, 0)$ have purely imaginary eigenvalues of $\lambda_{1,2} = \pm\sqrt{2}\, i$ and are hence centers (*i.e.*, elliptic points). A phase portrait for the unforced Duffing system is shown in Figure A.9. Note that the trajectories shown are actually streamlines for the vector field $\boldsymbol{f}(\boldsymbol{x})$. This equivalence between trajectories and streamlines holds for all autonomous systems.

There is much more to the classification of fixed points than we can cover here, and the interested reader is referred to standard texts on dynamical systems [17, 1, 13, 39, 35]. Nevertheless, the basic principles covered above regarding fixed points and their classification through linearization are critical concepts encountered over and over in the dynamical systems literature. In fact, we will revisit linearization

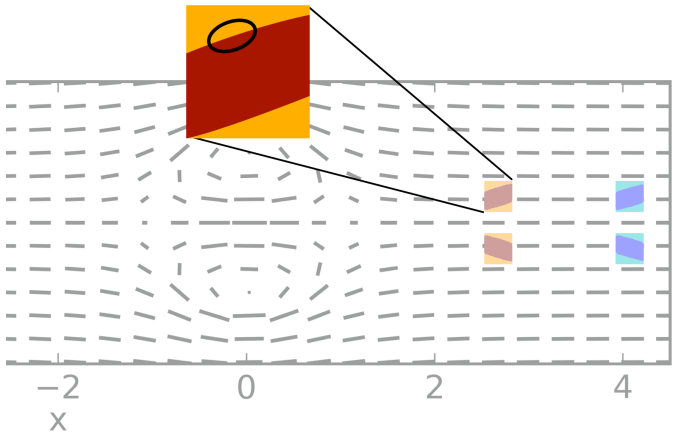

Figure A.10: Section of Figure A.5(a) with one of the orange/red blocks magnified in the inset. The group of tracers bounded by the ellipse in the inset are initially very close together, but evolve differently as shown in Figure A.5.

in the next section to extract essential information regarding long term behavior of arbitrary trajectories.

A.4 Lyapunov exponents

As we saw in the previous section, the way trajectories behave in the vicinity of a fixed point and the rate at which they move toward or away from the point can provide a great deal of insight into the underlying dynamics of a system. However the linearization of Section A.3 is generally only valid within some small region of the fixed point. After a sufficiently long period of time, it is not unreasonable to expect that $\delta\boldsymbol{x}$ will grow so large that the linearized system of Eq. A.12 is no longer valid. Hence we would like to extend the notion of monitoring how nearby trajectories separate to arbitrary trajectories that can be followed for long times.

Reflect for a moment on the perturbed Hill's spherical vortex in Section A.1 and the passive tracer field of Figure A.5. In particular, focus on the evolution of a group of tracers near the orange/red boundary as depicted in Figure A.10. This group of

tracers initially moves as a unit until they reach the rear of the vortex (at roughly $z = -1$ in Figure A.5(b)). Afterwards the orange tracers continue to move together, but the entire lot separates from the red. Two important notions are illustrated by this particular example.

First, the orange and red tracers belong to regions of the flow field that have qualitatively different dynamical behavior. Although the different color tracers start out close to one another, they eventually pull apart by color and evolve in disparate ways. That we do not immediately see the orange and red tracers behaving differently brings us to the second concept demonstrated by the Hill's spherical vortex example. In general, we have to follow the evolution of a system for a sufficiently long time, ideally as $t \to \infty$, to truly ascertain the dynamics. Just because two nearby trajectories remain close for some time does not mean they always will. Similarly two trajectories that initially separate may only do so momentarily.

We can mathematically express the rate at which arbitrary but nearby trajectories separate in the infinite time limit via Lyapunov exponents. To begin the development, let $\boldsymbol{F}(\boldsymbol{x}_0, t)$ represent a function that generates valid trajectories of Eq. A.7 based on initial conditions, $\boldsymbol{x}_0$. That is, $\boldsymbol{x}(t) = \boldsymbol{F}(\boldsymbol{x}_0, t)$ and $\boldsymbol{x}_0 = \boldsymbol{F}(\boldsymbol{x}_0, 0)$. The function $\boldsymbol{F}(\boldsymbol{x}_0, t)$ is known as the *flow* or *time*-t *map* and is simply the general solution of Eq. A.7 evaluated at the arbitrary initial conditions $\boldsymbol{x}_0$. Hence by constructing $\boldsymbol{F}(\boldsymbol{x}_0, t)$, we can form any single trajectory by fixing $\boldsymbol{x}_0$ and allowing t to vary. On the other hand, we can evaluate $\boldsymbol{F}(\boldsymbol{x}_0, t)$ at some particular fixed time, t_p, while varying $\boldsymbol{x}_0$ to map all possible initial conditions to their evolved locations at t_p.

Using the flow $\boldsymbol{F}$, a trajectory $\boldsymbol{x}_b$ that is initially close to another trajectory $\boldsymbol{x}_a$ can then be written as

$$\boldsymbol{x}_b(t) = \boldsymbol{F}(\boldsymbol{x}_b(0), t) = \boldsymbol{F}(\boldsymbol{x}_a(0), t) + \boldsymbol{\nabla}_0 \boldsymbol{F}|_{\boldsymbol{x}_a(0)} \delta\boldsymbol{x}_0 + O(\delta\boldsymbol{x}_0^2) \tag{A.18}$$

where we have linearized $\boldsymbol{F}$ about $\boldsymbol{x}_a(0)$. Note that $\delta\boldsymbol{x}_0 = \boldsymbol{x}_b(0) - \boldsymbol{x}_a(0)$. Furthermore, the subscript 0 on the gradient operator is merely used to emphasize that the linearization is with respect to the dependence of $\boldsymbol{F}$ on initial conditions. Neglecting the higher order terms, setting $\delta\boldsymbol{x}(t) = \boldsymbol{x}_b(t) - \boldsymbol{x}_a(t)$, and recognizing that $\boldsymbol{x}_a(t) = \boldsymbol{F}(\boldsymbol{x}_a(0), t)$, we then have

$$\delta\boldsymbol{x}(t) = \boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)}\delta\boldsymbol{x}_0. \tag{A.19}$$

Since the Taylor series expansion of $\boldsymbol{F}$ is about the initial conditions $\boldsymbol{x}_a(0)$, Eq. A.19 is valid for all time provided that we can justify dropping the higher order terms from Eq. A.18. These terms can in general be safely ignored as long $\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)} \neq 0$ and the initial separation of the trajectories, $\delta\boldsymbol{x}_0$, is sufficiently small. While the second condition is clear, the first deserves a comment. When $\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)} = 0$, the higher order terms of the expansion must be considered in turn as one of them may be nonzero and hence dominate the variation of $\boldsymbol{F}$ on initial conditions. For our purposes, we will simply assume $\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)} \neq 0$. In what follows, we will drop evaluation of $\boldsymbol{\nabla}_0\boldsymbol{F}$ at $\boldsymbol{x}_a(0)$ from the notation, but we should keep in mind that the linearization of $\boldsymbol{F}$ was about the initial condition $\boldsymbol{x}_a(0)$. Consequently, the following results can be expected to vary with the choice of $\boldsymbol{x}_a(0)$.

Let us now focus on how nearby trajectories move apart and not how their relative orientation varies with time. Using Eq. A.19, we can compute the change in length of $\delta\boldsymbol{x}$ as

$$\zeta(t) = \frac{\|\delta\boldsymbol{x}(t)\|}{\|\delta\boldsymbol{x}_0\|} = \frac{\sqrt{\delta\boldsymbol{x}_0^T\left(\boldsymbol{\nabla}_0\boldsymbol{F}\right)^T\boldsymbol{\nabla}_0\boldsymbol{F}\delta\boldsymbol{x}_0}}{\|\delta\boldsymbol{x}_0\|} = \frac{\sqrt{\delta\boldsymbol{x}_0^T\boldsymbol{M}\delta\boldsymbol{x}_0}}{\|\delta\boldsymbol{x}_0\|} \tag{A.20}$$

where the superscript T denotes the standard transpose and

$$\boldsymbol{M} = \boldsymbol{M}(t) \equiv \left(\boldsymbol{\nabla}_0\boldsymbol{F}\right)^T\boldsymbol{\nabla}_0\boldsymbol{F}. \tag{A.21}$$

Let $\{\lambda_i = \lambda_i(t)\}_{i\in(1,n)}$ be the n eigenvalues of $\boldsymbol{M}$ arranged in order of decreasing magnitude, and $\{\boldsymbol{v}_i = \boldsymbol{v}_i(t)\}_{i\in(1,n)}$ the corresponding normalized eigenvectors. Since $\boldsymbol{M}$ is a symmetric matrix, its eigenvectors are orthogonal. As a result, $\zeta(t)$ will be a maximum when $\delta\boldsymbol{x}_0$ is aligned with $\boldsymbol{v}_1$ and similarly a minimum when aligned with $\boldsymbol{v}_n$. Let $\delta\bar{\boldsymbol{x}}_0$ be an initial separation vector chosen parallel to an eigenvector $\boldsymbol{v}_i$ of $\boldsymbol{M}$, then for each eigendirection we have

$$\zeta_i(t) = \frac{\sqrt{\delta\bar{\boldsymbol{x}}_0^T \boldsymbol{M} \delta\bar{\boldsymbol{x}}_0}}{\|\delta\bar{\boldsymbol{x}}_0\|} = \frac{\sqrt{\delta\bar{\boldsymbol{x}}_0^T \lambda_i \delta\bar{\boldsymbol{x}}_0}}{\|\delta\bar{\boldsymbol{x}}_0\|} = \sqrt{\lambda_i}, \quad i = 1, \ldots, n. \tag{A.22}$$

From our requirement that the dynamical system be Lipschitz continuous, we know via Eq. A.9 that $\zeta(t)$ can vary at most exponentially. For each eigendirection of $\boldsymbol{M}$, we can therefore write

$$\zeta_i(t) = \sqrt{\lambda_i} \leq e^{K_i t}. \tag{A.23}$$

Given that the separation between nearby trajectories varies at most exponentially with time, simply assume that the rate of separation change is always exponential. A corresponding value of K_i along each eigendirection can then be computed that produces the correct observed separation at time t as though the distance between trajectories did indeed evolve exponentially. Consequently we have via Eq. A.23

$$K_i = \frac{1}{2t} \ln \lambda_i. \tag{A.24}$$

Which in the long-time limit yields

$$\sigma_i = \lim_{t\to\infty} K_i = \lim_{t\to\infty} \frac{1}{2t} \ln \lambda_i, \quad i = 1, \ldots, n \tag{A.25}$$

Since the linearization of Eq. A.19 was about $\boldsymbol{x}_a(0)$, the σ_i are known as the *Lyapunov exponents* of the trajectory $\boldsymbol{x}_a$, and they provide a measure of the rate at which other nearby trajectories are attracted to or repelled from $\boldsymbol{x}_a$ along the n orthogonal

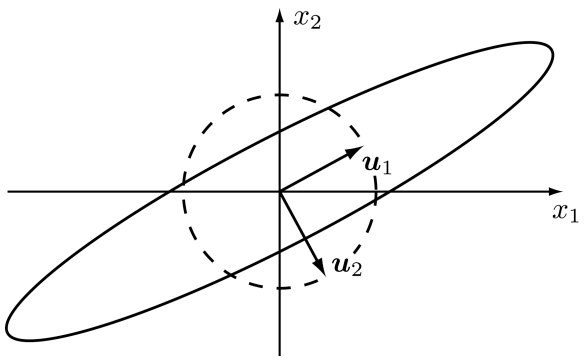

Figure A.11: In 2D, the matrix $\boldsymbol{\nabla}_0\boldsymbol{F}$ transforms the unit circle (dashed line) into an ellipse whose axes align with the unit vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. These vectors are the normalized eigenvectors of the matrix $\boldsymbol{N}$ (see text).

eigenvectors, $\boldsymbol{v}_i$, of $\boldsymbol{M}$.

While the eigenvectors of $\boldsymbol{M}$ permit us to determine what orientations of $\delta\boldsymbol{x}_0$ produce maximal or minimal stretching, they do not necessarily tell us the direction in which the duly stretched or shrunken $\delta\boldsymbol{x}$ points. If we were to release a ball of initial conditions around $\boldsymbol{x}_a(0)$ and watch its evolution, we would find that the ball is possibly translated along the trajectory $\boldsymbol{x}_a$, but also stretched into an ellipsoid. This mapping of an n-dimensional sphere onto a corresponding ellipsoid is characteristic behavior of any $n\times n$ invertible matrix [34], and $\boldsymbol{\nabla}_0\boldsymbol{F}$ is no exception. The orthogonal axes of the ellipsoid generated by the action of $\boldsymbol{\nabla}_0\boldsymbol{F}$ on the ball of initial conditions align with the normalized eigenvectors $\{\boldsymbol{u}_i = \boldsymbol{u}_i(t)\}_{i\in(1,n)}$ of the matrix $\boldsymbol{N} = \boldsymbol{N}(t) \equiv \boldsymbol{\nabla}_0\boldsymbol{F}\left(\boldsymbol{\nabla}_0\boldsymbol{F}\right)^T$, as is demonstrated in Figure A.11.[9]

It is not uncommon to find Lyapunov exponents developed from the gradient of the vector field $\boldsymbol{f}(\boldsymbol{x})$ [13, 7, 11, 39], however the approach is equivalent to that given above. To see this, we begin with Eq. A.19 and determine the differential equation $\delta\boldsymbol{x}(t)$ satisfies. Note that in general the linearized $\delta\boldsymbol{x}(t)$ will not be a solution to the autonomous system of Eq. A.7. Differentiation of Eq. A.19 with respect to time

[9]For more on how general matrices deform the unit sphere, refer to a discussion of the singular value decomposition (SVD) found in any standard text on linear algebra [34, 33].

provides

$$\delta\dot{\boldsymbol{x}} = \frac{d}{dt}\left(\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)}\right)\delta\boldsymbol{x}_0. \tag{A.26}$$

Since $\boldsymbol{F}(\boldsymbol{x}_0, t)$ is a valid solution to the autonomous system, Eq. A.7 can be rewritten as

$$\frac{d}{dt}\boldsymbol{F} = \boldsymbol{f}(\boldsymbol{F}), \tag{A.27}$$

and taking the gradient with respect to initial conditions of both sides yields

$$\frac{d}{dt}\boldsymbol{\nabla}_0\boldsymbol{F} = \boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{f}|_{\boldsymbol{F}}\,\boldsymbol{\nabla}_0\boldsymbol{F} \tag{A.28}$$

via the chain rule. Substituting Eq. A.28 into A.26 and recognizing that by construction $\delta\boldsymbol{x} = \boldsymbol{\nabla}_0\boldsymbol{F}\,\delta\boldsymbol{x}_0$ we find

$$\delta\dot{\boldsymbol{x}} = \boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{f}|_{\boldsymbol{x}_a}\,\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)}\,\delta\boldsymbol{x}_0 = \boldsymbol{\nabla}\boldsymbol{f}|_{\boldsymbol{x}_a}\,\delta\boldsymbol{x}. \tag{A.29}$$

Eq. A.29 is the differential equation satisfied by Eq. A.19, and it is also clearly a linearization of the autonomous system about the trajectory $\boldsymbol{x}_a(t)$. Given that Eq. A.19 satisfies Eq. A.29, the matrix $\boldsymbol{\nabla}_0\boldsymbol{F}|_{\boldsymbol{x}_a(0)}$ is the *fundamental solution matrix* of the linearized system.[10] Note that $\boldsymbol{\nabla}\boldsymbol{f}|_{\boldsymbol{x}_a}$ is generally a function of time, since the velocity gradient tensor is being evaluated along the trajectory $\boldsymbol{x}_a(t)$. And therein lies the essential difference between the present linearization and that used during the classification of fixed points in Section A.3 — here we have permitted the trajectory about which the linearization was performed to vary with time. Maintaining time dependence in the Jacobian of the velocity field for Eq. A.29 can be understood by reflecting on the long-time validity of its solution, Eq. A.19, as discussed earlier. While we certainly could evaluate $\boldsymbol{\nabla}\boldsymbol{f}$ in Eq. A.29 at some fixed location along a given

[10] A fundamental solution matrix transforms initial conditions into a solution at time t. That is a solution $\boldsymbol{\phi}(t)$ to some system of differential equations can be written as $\boldsymbol{\phi}(t) = \boldsymbol{\Phi}(\boldsymbol{\phi}_0, t)\boldsymbol{\phi}_0$, where $\boldsymbol{\Phi}(\boldsymbol{\phi}_0, t)$ is the fundamental solution matrix.

trajectory, for a general dynamical system both trajectories defining $\delta\boldsymbol{x}$ would likely move far enough away from the chosen stationary position to make the linearization invalid. Only by following one of the trajectories through time, in this case $\boldsymbol{x}_a$, can we construct a linearized system of differential equations having the same long-time legitimacy as Eq. A.19. On the other hand, if $\boldsymbol{x}_a$ happens to be a fixed point of the autonomous system, then the linearization of A.29 is identical to that of Section A.3.

For an example regarding the computation of Lyapunov exponents, consider the nonlinear system (ref. [39], Example 29.2.3)

$$\dot{\boldsymbol{x}} = \begin{pmatrix} x - x^3 \\ -y \end{pmatrix}, \tag{A.30}$$

subject to the initial conditions $\boldsymbol{x}(0) = (x_0, y_0)$. The flow $\boldsymbol{F}$ is given by

$$\boldsymbol{F}(\boldsymbol{x}_0, t) = \begin{pmatrix} x_0 e^t/\sqrt{\beta} \\ y_0 e^{-t} \end{pmatrix}, \quad x_0 \neq \pm 1, \tag{A.31}$$

where $\beta = 1 + x_0^2(e^{2t} - 1)$. The matrix $\boldsymbol{M}$ becomes

$$\boldsymbol{M} = (\boldsymbol{\nabla}_0 \boldsymbol{F})^T \boldsymbol{\nabla}_0 \boldsymbol{F} = \begin{pmatrix} \beta^{-3} e^{2t} & 0 \\ 0 & e^{-2t} \end{pmatrix}, \tag{A.32}$$

and the eigenvalues of $\boldsymbol{M}$ can be obtained from the characteristic equation

$$\lambda^2 - \lambda \left(\beta^{-3} e^{2t} + e^{-2t} \right) + \beta^{-3} = 0. \tag{A.33}$$

Note that the dependence of $\boldsymbol{M}$ on initial conditions $\boldsymbol{x}_0$ is limited to the factor β which in turn only depends on x_0. To determine the Lyapunov exponents for the fixed point at the origin, we see that $\beta = 1$ when $x_0 = 0$. Solving Eq. A.33 for λ then

provides

$$\lambda_{1,2} = \frac{\gamma \pm \sqrt{\gamma^2 - 4}}{2}, \quad x_0 = 0 \tag{A.34}$$

where $\gamma = e^{2t} + e^{-2t}$. In the limit that $t \to \infty$, $\sqrt{\gamma^2 - 4}$ varies roughly as e^{2t} and hence Eq. A.34 behaves as

$$\lambda_{1,2}(t) \sim \gamma \pm e^{2t} = e^{2t} + e^{-2t} \pm e^{2t} = e^{2t}, e^{-2t} \quad \text{as } t \to \infty. \tag{A.35}$$

By Eq. A.25, the Lyapunov exponents for the fixed point at the origin are then

$$\sigma_{1,2} = \pm 1. \tag{A.36}$$

The brief treatment of Lyapunov exponents above captures just some of the essentials. For more on the topic, see [26, 1] and the references therein. What we have seen here, however, is that the long term behavior of two nearby trajectories can be characterized using Lyapunov exponents. These exponents provide a measure of the rate at which nearby trajectories separate in the limit $t \to \infty$. We have chosen to mathematically develop the Lyapunov exponents using the flow, $\boldsymbol{F}(\boldsymbol{x}_0, t)$, since $\boldsymbol{F}$ is specialized to a particular trajectory only through the initial conditions $\boldsymbol{x}_0$. Consequently it is much easier to envision how the ensuing linearization of $\boldsymbol{F}$ constructed to derive the Lyapunov exponents can be valid for long times — the linearization is with respect to initial conditions alone. Nevertheless, we also have shown how the Lyapunov exponents can be equivalently derived from the gradient of the autonomous system's vector field, $\boldsymbol{f}(\boldsymbol{x})$. At this point it should be conceptually clear how a positive Lyapunov exponent indicates sensitive dependence on initial conditions. Namely, if the orientation of the initial separation between the trajectories is chosen appropriately, a positive Lyapunov exponent indicates the trajectories will pull apart exponentially with time.

A.5 Stable and unstable manifolds

We now come to the principal dynamical systems concept of this document: stable and unstable manifolds of state space structures. First a note on terminology. A *manifold* is a space that locally resembles Euclidean space such that we can in some neighborhood use all the familiar tools of Euclidean geometry. Ignoring topography, the spherical surface of Earth on a small enough scale looks like a Euclidean plane,[11] and we can construct buildings, roads, and airports using right angles and traditional Cartesian coordinates. The letter O, by comparison, resembles a line when viewed on a sufficiently small scale. Hence a manifold is simply an abstraction of spatial structures that topologically resemble Euclidean equivalents.

Suppose we have a system with a stable node, then as we discussed above, trajectories within some neighborhood of the point will decay to the fixed point as $t \to \infty$. Approaching trajectories can only reach the node in the infinite time limit, otherwise we would have the crossing of two trajectories, the fixed point and a decaying trajectory, which violates the uniqueness of solution theorem. Nevertheless the set of all trajectories that can reach any type of fixed point in the forward infinite time limit constitutes the *stable manifold* of the fixed point. Furthermore, the concept of a stable manifold is not limited to fixed points, for the set of trajectories that decay to any state space structure can be classified as the stable manifold of the structure. However to be part of the stable manifold, it is not sufficient for a trajectory to simply approach a particular structure, since at some later time the same trajectory could begin moving away. Instead, the stable manifold is composed of those very special trajectories for which the system state asymptotically converges to that of a specific structure (*i.e.*, stable manifold trajectories reach the structure, but only in the infinite time limit).

We have to be more careful when defining the unstable manifold, or we could end

[11] Obviously to such a degree that many ancient cultures believed Earth was flat.

up erroneously counting any trajectory which moves away from a state space structure of interest as part of the manifold. The trick for the unstable manifold is recognition that of all the trajectories that may be fleeing a structure, only a special subset may asymptotically converge to it as $t \to -\infty$. If there are trajectories that reach a structure in the negative infinite time limit, then those trajectories constitute the *unstable manifold* of the structure. So in summary, the stable and unstable manifolds of a fixed point or other structure are the set of trajectories that converge to the feature in the forward or backward infinite time limit, respectively.

Manifolds are collections of trajectories, and the trajectories that make up a manifold form an *invariant set*. That is members of the set can never escape from it, and the set never admits new members. Since manifolds are invariant sets, they form impenetrable barriers to other trajectories that are not part of the manifold. Consequently stable and unstable manifolds serve to organize state space by trapping groups of trajectories within the confines of the manifolds. Figure A.12 shows phase portraits for the steady Duffing and Hill's spherical vortex systems (Eq.'s A.1 and A.2, respectively, with $\epsilon = 0$). The manifolds of the hyperbolic points are highlighted. Notice that the stable and unstable manifolds of the Duffing system's saddle point are joined to form a *homoclinic connection* that surrounds each of the elliptic points. In this manner, the manifolds act as separatrices delimiting zones of the flow field with very different behavior.

As for Hill's spherical vortex, we see that stable and unstable manifolds of two separate hyperbolic points form two *heteroclinic connections* — one along the z-axis spanning $-1 < z < 1$, and the other curved to enclose the recirculation zone. Keep in mind that Hill's spherical vortex is a 3D flow, and that we are viewing trajectories in a symmetry plane for convenience. The outer, curved heteroclinic connection is actually the surface of a sphere and encloses a doughnut-shaped recirculation zone in 3D. These outer, spherical manifolds of the vortex form separatrices in a manner analogous to

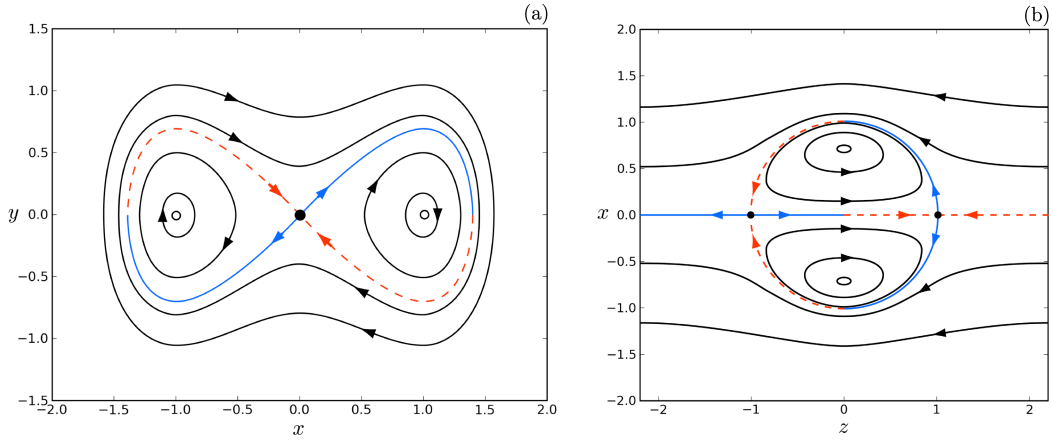

Figure A.12: Phase portraits of the steady (a) Duffing oscillator and (b) Hill's spherical vortex. Hyperbolic fixed points are denoted by solid, black circles. Stable manifolds of the hyperbolic points are shown as dashed, red lines, and unstable manifolds are presented as solid blue lines. Several other trajectories are shown for reference as black lines.

that of the Duffing oscillator by marking a transition between the recirculating flow of the vortex and the free-stream ambient fluid passing over the flow structure.

To determine how manifolds are oriented in the near vicinity of a fixed point, for example, we once again turn to linearization. As we saw previously, the general solution, Eq. A.13, of the linearized system (Eq. A.12) is composed of n, linearly independent functions, each of which is also a solution. The component solutions have a characteristic speed given by the eigenvalues of $\boldsymbol{J}$ and a characteristic direction. For systems where $\boldsymbol{J}$ has a full set of n, linearly independent eigenvalues, the characteristic direction of each component solution is provided by the corresponding eigenvector. If $\boldsymbol{J}$ does not have a full complement of eigenvectors, then some of the component solutions will align with the available eigenvectors, and the others will align with characteristic directions provided by what are known as *generalized eigenvectors* [5]. For simplicity, temporarily consider the case where $\boldsymbol{J}$ does indeed have a full set of

eigenvectors thereby producing a general solution of the form

$$\delta\boldsymbol{x}(t) = c_1\boldsymbol{v}_1 e^{\lambda_1 t} + c_2\boldsymbol{v}_2 e^{\lambda_2 t} + \ldots + c_n\boldsymbol{v}_n e^{\lambda_n t}, \quad i = 1, \ldots, n \tag{A.37}$$

with the eigenvalues of $\boldsymbol{J}$ represented by λ_i. If a perturbed trajectory of the linearized system is a) parallel to a particular eigenvector, $\boldsymbol{v}_p$, with $\mathrm{Re}(\lambda_p) \neq 0$, and b) intersects the fixed point (in an infinite time limit), then all of the other coefficients of integration in Eq. A.37 (*i.e.*, the c_i) will be zero except for the component solution that aligns with $\boldsymbol{v}_p$. In this case $\delta\boldsymbol{x}$ has the form

$$\delta\boldsymbol{x}(t) = c_p\boldsymbol{v}_p e^{\lambda_p t} \tag{A.38}$$

and the perturbed trajectory will asymptotically approach the fixed point along the eigenvector in the forward or backward infinite time limit depending on the sign of λ_p. But any trajectory that intersects a fixed point in an infinite time limit is a member of the stable or unstable manifold of that point. Therefore the eigenvectors or generalized eigenvectors from the linearized system of equations, Eq. A.12, are tangent to the stable and unstable manifolds of the autonomous system, Eq. A.7. This observation is simply an informal statement of the stable manifold theorem (ref., [13, Thrms. 1.3.2 and 5.2.8]), and the concept is readily extended to other state space structures.

Given that manifolds are collections of trajectories and not trajectories themselves, they can intersect transversely under special circumstances.[12] The first principal requirement being that state space must have at least three dimensions, either naturally for an autonomous system or by converting a non-autonomous system to an autonomous equivalent as discussed in Section A.3. Additionally, manifolds of

[12]If the angle between lines tangent to the intersecting manifolds is nonzero, then the intersection is transverse. The letter x, for example, could be viewed as the transverse intersection of two line segments.

the same stability type (*e.g.*, the stable manifolds of a saddle point) cannot intersect transversely [13], leaving the only possible transverse intersections as being of the stable-unstable type. Intersecting stable and unstable manifolds may seem to violate the prohibition against intersecting trajectories, but we must not forget that these manifolds are sets of trajectories, not trajectories themselves. The seam formed by such an intersection, however, is a trajectory that forever belongs to both the stable and unstable manifolds that have intersected. The trajectory of the intersection must twist around state space eternally receding from one parent structure (*e.g.*, a fixed point) while at the same time moving ever closer to the same or other structure. If the intersecting manifolds originate from a single state space feature, the trajectory of the intersection is known as a *homoclinic trajectory*. On the other hand, transversely intersecting manifolds that belong to different structures form a *heteroclinic trajectory* at the seam. These concepts are presented in Figure A.13 via two *Poincaré sections*.[13]

If a dynamical system is bounded[14] and time invariant or exhibits periodic behavior, then a transverse intersection of the stable and unstable manifolds has profound consequences. In bounded systems, the homoclinic or heteroclinic trajectories must unendingly trace an infinite tangle of manifold intersections as the trajectory navigates a state space of finite extent. For a periodic system, the tangle can be spread out over an infinitely large state space, but the periodicity ties the far flung sections of the tangle together by mapping one region to another. Regardless of the system type, tangles are one of the harbingers of chaotic behavior — any non-manifold trajectory that happens to be trapped by the intertwined, impenetrable manifolds will usually have to twist around state space eternally caught in the confines of the ravel. The two

[13]A Poincaré section is merely the name given to a special cross-section constructed to aid analysis of higher dimension dynamical systems. The section is formed by recording the intersection of one or more trajectories crossing a plane in a chosen direction. For more on Poincaré sections, see [27], or for a very visual example of Poincaré sections derived from a real fluid flow, see [9, 8].

[14]Recall that a bounded system's trajectories are confined to a finite region of state space but not necessarily closed. That is, a bounded trajectory does not fly off to infinity.

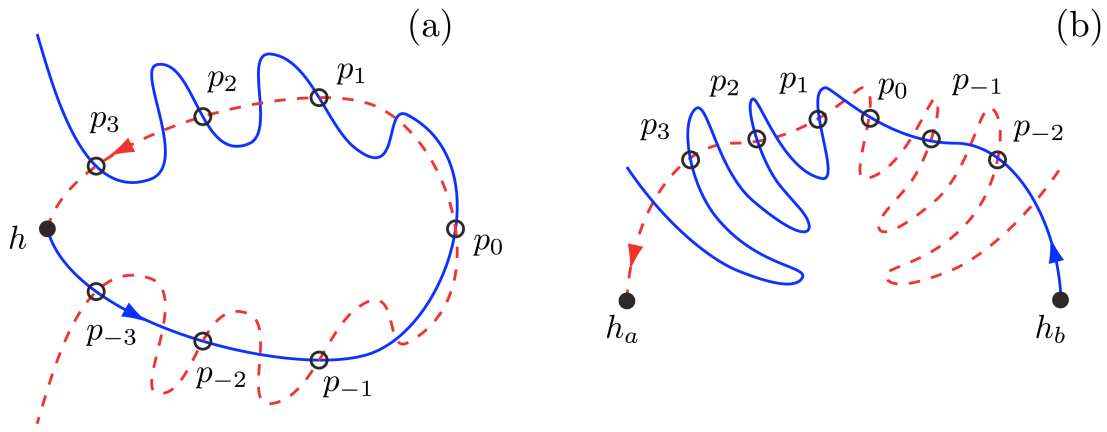

Figure A.13: Homoclinic (a) and heteroclinic (b) manifold tangles illustrated on a plane that cuts through the actual 3D manifolds. The hyperbolic points, shown as solid black circles, mark the intersection of a closed trajectory (*e.g.*, a saddle cycle) in 3D state space with the plane. Similarly, the intersections of the (a) homoclinic and (b) heteroclinic trajectories with the plane of section are depicted by open circles. The subscript denotes the order in which the intersections occur relative to p_0.

example systems from Section A.1 provide excellent case studies of the complexity of manifold tangles and the resulting impact such features have on system dynamics.

Let us begin with the forced Duffing system having $\epsilon = 0.5$ and $\omega = 3$. This non-autonomous system is now an explicit function of time and must be converted to autonomous form so that the tools of dynamical systems theory are more readily applied. As discussed in Section A.3, the equivalent autonomous system can be written

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= x - x^3 + \epsilon \cos\theta \\ \dot{\theta} &= \omega. \end{aligned} \tag{A.39}$$

Notice that the system and extended state space are now three-dimensional which should intuitively make sense given that we need three pieces of information to fully specify the state of the system, x, y, and t. But we also are dealing with a periodic

system where the forcing term takes unique values only for $0 \leq \theta < 2\pi$. Consequently, the extended state space can be envisioned as a doughnut created by revolving the xy-plane about some line parallel to the y-axis. The θ-axis, corresponding to time, is then a circle.

Note that the system of A.39 has no fixed points for the constant $\omega \neq 0$. By forcing the system with a time dependent function, we lose all fixed points, but we typically have other distinguished hyperbolic trajectories that remain [19, 21]. Thus when we perturb the Duffing system, the saddle fixed point of the steady system with $\epsilon = 0$ is transformed into a saddle cycle — a closed trajectory to which other trajectories are attracted and repelled along certain directions. We can envision the saddle cycle as being a ring of saddle points in the extended state space of the forced system. Perturbing the Duffing oscillator has another very import effect, however. Namely, the homoclinic connection that originally existed between the stable and unstable manifolds of the saddle point shown in Figure A.12(a) breaks, and a homoclinic tangle is formed. It is the stable and unstable manifolds of the saddle cycle that intersect transversely when the system is perturbed. The geometry of the intersecting manifolds in the extended, 3D state space is illustrated in Figure A.14, but only a very small segment of the manifolds can be presented since they are infinitely long. The resulting tangle is, however, extraordinarily complex, and Figure A.15 provides a cross-section of slightly longer manifold segments to further emphasize the intricacy.

The effect of the manifold tangle on a packet of fluid released within the zone of intersections is to repeatedly stretch and fold the material as time evolves. The origin of such behavior can be conceptually understood by reflecting on Figures A.14 and A.15. In particular, note how the manifold intersections form small impenetrable compartments in Figure A.14, each of which rotates around the toroidal tangle getting mapped to a new compartment every period of the forcing. Each period some of these tiny chambers map to new ones along one direction in the xy-plane while others

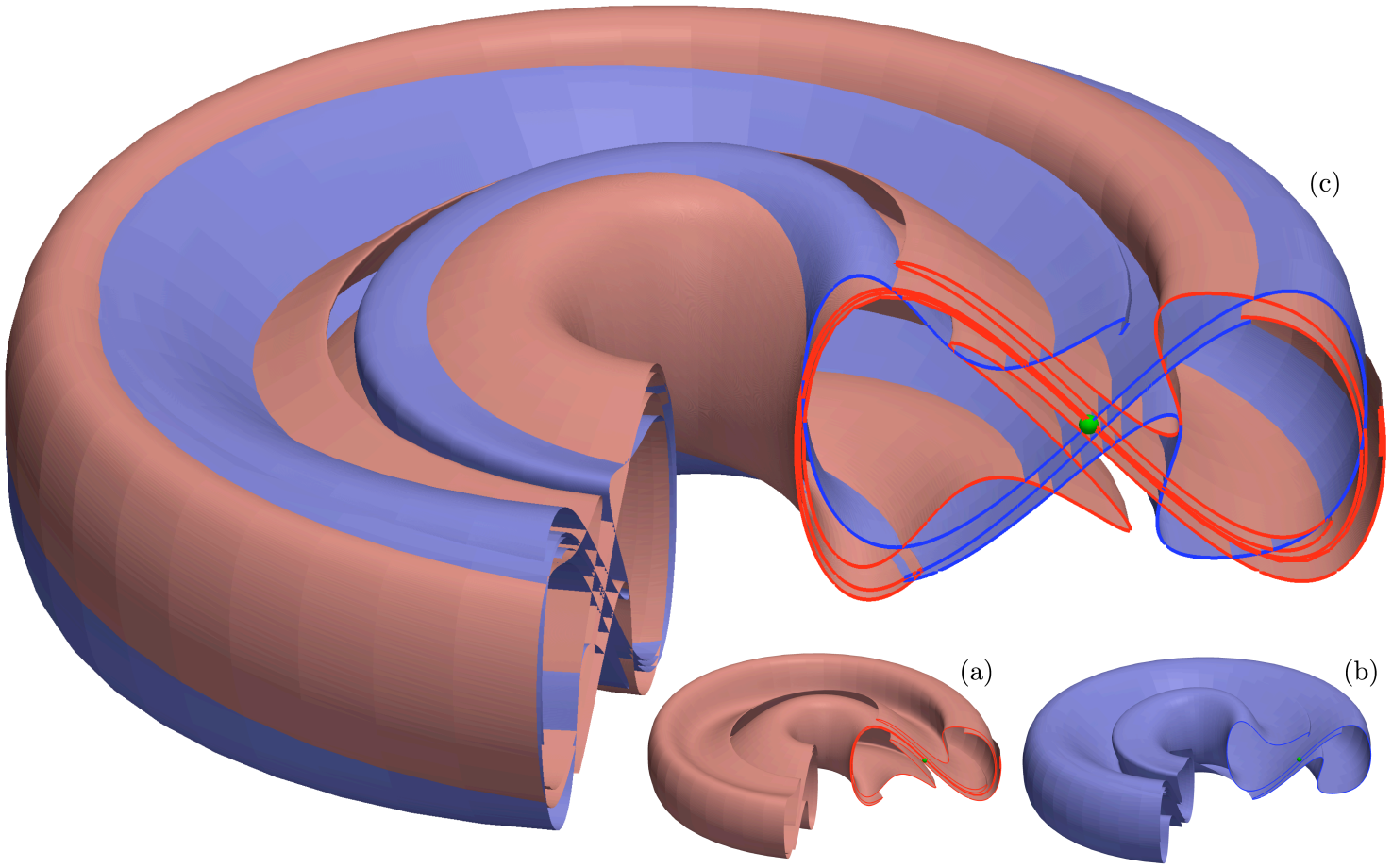

Figure A.14: Segments of stable and unstable manifolds for the forced Duffing system's saddle cycle. A portion of state space spanning $3\pi/2 \leq \theta < 2\pi$ has been removed to show the complicated tangle that has formed. The intersection of the manifolds with the $\theta = 0$ plane is highlighted, and the corresponding saddle cycle intersection is denoted by the green ball. The stable manifold (red) is shown alone in the inset (a). Similarly, the unstable manifold (blue) is shown inset in (b). The combined structures are presented in (c).

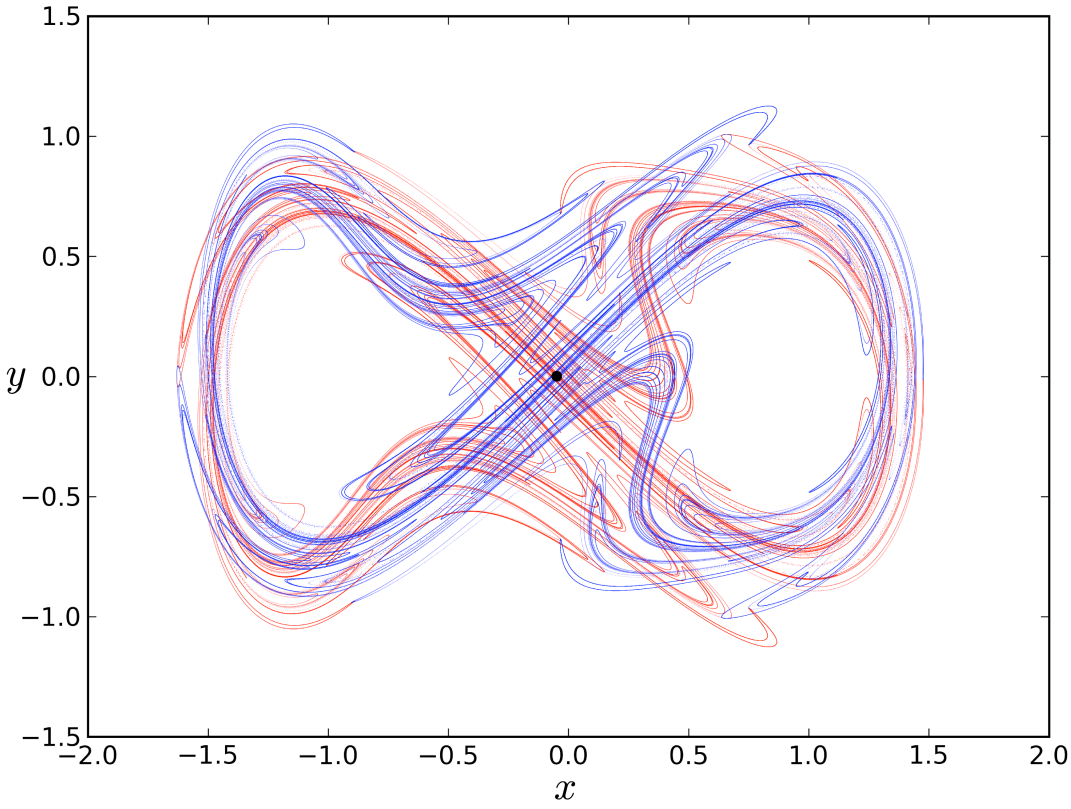

Figure A.15: Longer segments of the stable (red) and unstable (blue) manifolds shown intersecting the $\theta = 0$ plane. The saddle cycle intersection is represented by the solid black circle near the origin.

map along different directions. The end result is that the fluid we view in the xy-plane stretches and folds again and again. The dynamics associated with a tangling of stable and unstable manifolds is responsible for the efficient stirring observed in Figures A.2(b,f). In both cases, we can now see that the blocks of tracers were initially released into a region of the flow field occupied by the homoclinic tangle of the saddle cycle's manifolds. Indeed, the figures of the stirred fluid closely resemble slightly longer sections of the unstable manifold shown in Figure A.15.

Further inspection of Figures A.14 and A.15 also indicates that while the geometry of the intersecting manifolds is extremely intricate and fills a large part of state space, there are two enclosed areas approximately centered at $(x, y) = (\pm 1, 0)$ which the saddle cycle manifolds apparently cannot penetrate. These regions roughly align with the original rotational flow zones of Figure A.12(a), and are in fact the remnants of those unperturbed features. Some of the original closed trajectories do in a sense remain. In the perturbed system, however, these persisting trajectories may no longer be closed, and they certainly no longer circle around the xy-plane alone. Remember that in the extended state space of the perturbed system, the θ-axis forms a circle, so trajectories spiral around the θ-axis as they evolve. Consequently those original closed trajectories of the unperturbed system that remain, now travel around the surface of a torus in the extended state space of the forced system. Just like stable and unstable manifolds, some of these tori form invariant, impenetrable surfaces that trap trajectories within their bounds. Poincaré sections can be constructed that reveal the locations of tori and surrounding structures in the same fashion as has been done for the stable and unstable manifolds shown elsewhere. Figure A.16 shows such a Poincaré section again constructed at the $\theta = 0$ plane. Instead of following a large number of particles for a relatively short time, as was done elsewhere to grow the stable and unstable manifolds, this section has tracked a small number of particles for a very long time. The stable and unstable manifolds are no longer readily visible

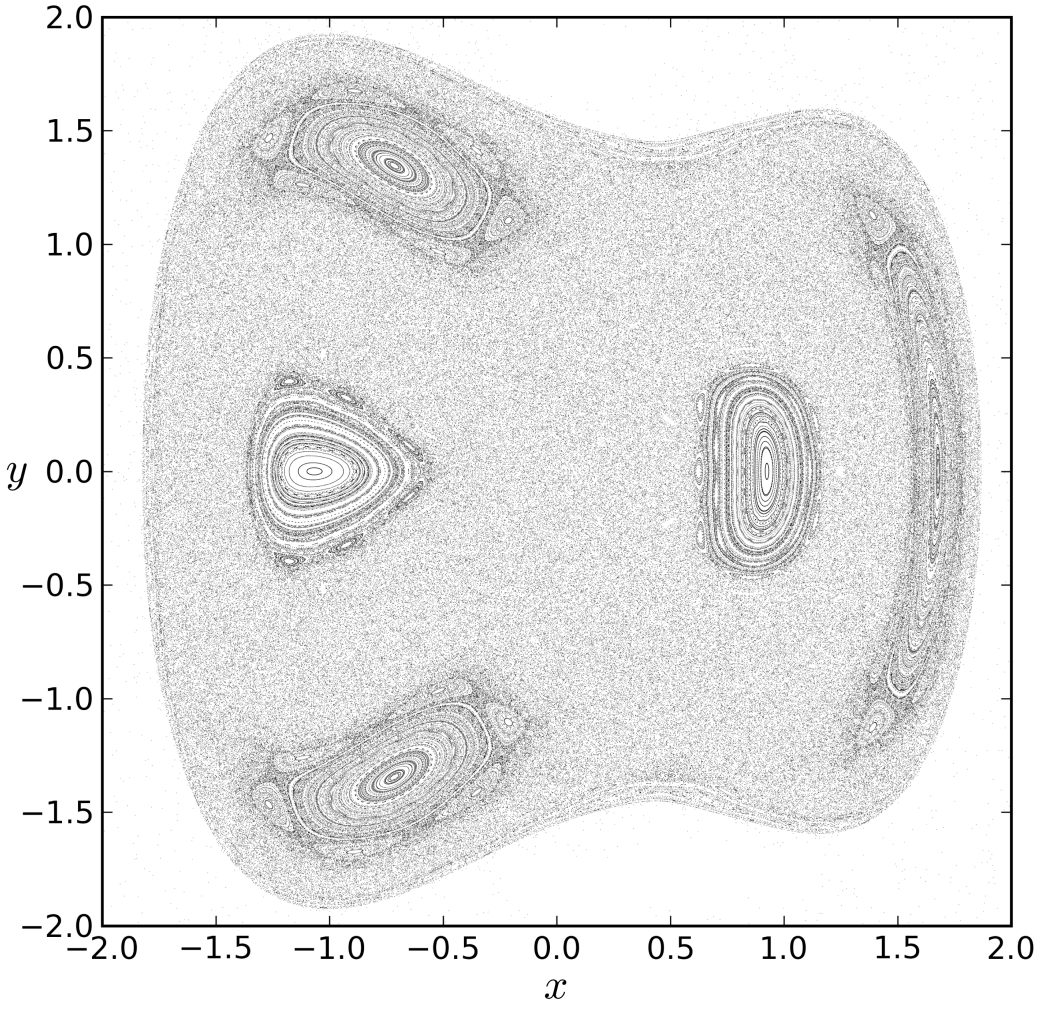

Figure A.16: Poincaré section for the forced Duffing oscillator at the $\theta = 0$ plane.

in Figure A.16 but instead are part of the chaotic background fuzz that resembles noise. Note the presence of a series of concentric rings located in the vicinity of $(x, y) = (\pm 1, 0)$. These are cross-sections of some of the tori mentioned earlier. As illustrated by Figure A.17, the tori that produce the rings at $(x, y) = (-1, 0)$ are not associated with the set that produces the rings at $(x, y) = (1, 0)$. Consequently any fluid released within either of these regions will remain isolated and confined to that region of the xy-plane, which explains the behavior observed in Figures A.2(c,d).

The Poincaré section of Figure A.16 shows three other sets of concentric rings along the perimeter of the guitar-shaped region of the section. These rings are not related to the original, unperturbed recirculation zones, but are nonetheless cross-sections of additional tori present in the perturbed system. Notice how the existence of these rings, roughly centered around $(x, y) = (-1, \pm 1.5), (1.5, 0)$, is in no way obvious from examining the velocity field of Figure A.1. We might have predicted that some remnant of the original recirculation zone would survive, but a complicated toroidal structure that pierces the plane of section exactly three times is quite unexpected. Nevertheless, any fluid released in one of these regions does remain isolated, but the material also traverses the xy-plane in a clockwise fashion as Figure A.17 illustrates. Each period of the forcing, the material in one of these zones gets mapped to its nearest clockwise neighbor. So the Poincaré section has also provided an explanation for the scenario of Figure A.3.[15]

Through the tools of dynamical systems theory, we can now conclusively predict where fluid must be located such that it stirs thoroughly under the action of the perturbed Duffing oscillator. If we release dyed fluid in any of the regions bounded by an invariant torus, the material will remain isolated. On the contrary, we can release two different colored blocks of fluid anywhere in the fuzzy region of Figure A.16, and regardless of where the blocks are located relative to one another, they will

[15]There is much more to the dynamics of invariant tori than we can possibly cover here. The interested reader is therefore referred to [13, 2, 27, 4, 29] and ensuing discussions.

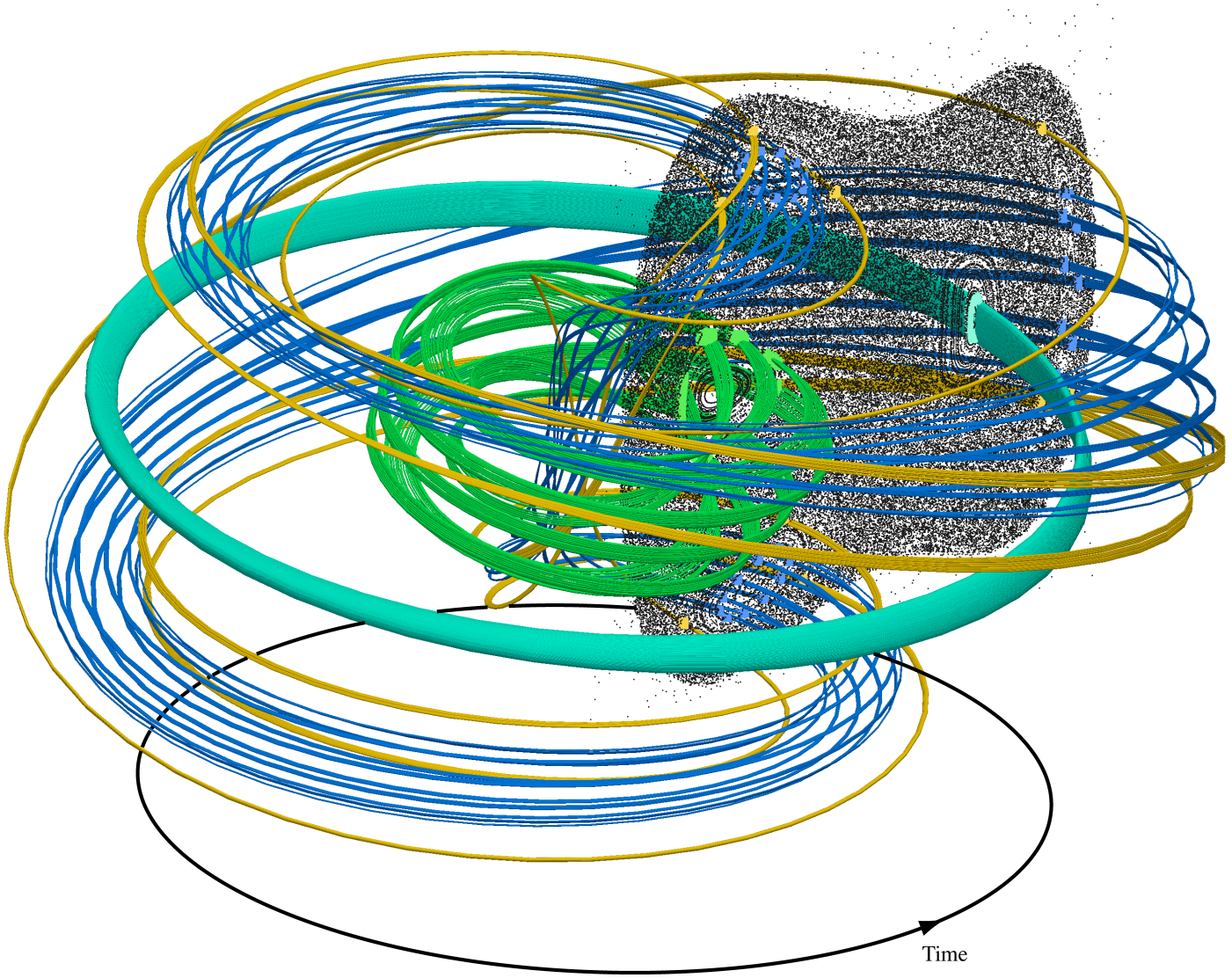

Figure A.17: Poincaré section of the forced Duffing oscillator shown in 3D state space along with several example trajectories, each assigned a unique color. The intersection of the trajectories with the plane of section ($\theta = 0$) has been highlighted for clarity.

stir thoroughly.

Analysis of the perturbed Hill's spherical vortex example of Section A.1 proceeds in a similar fashion. The equivalent autonomous system in extended state space becomes

$$\begin{aligned}
\dot{x} &= (u_r \sin\theta + u_\theta \cos\theta)\cos\phi - \frac{\epsilon}{2}\sin(\psi) \\
\dot{y} &= (u_r \sin\theta + u_\theta \cos\theta)\sin\phi - \frac{\epsilon}{2}\sin(\psi) \\
\dot{z} &= u_r \cos\theta - u_\theta \sin\theta + \epsilon \sin(\psi) \\
\dot{\psi} &= \omega
\end{aligned} \tag{A.40}$$

with u_r and u_θ provided by Eq.'s A.3 and A.4. As a reminder, we have chosen $a = 1$, $u_0 = -1.5$, $\epsilon = 0.3$, and $\omega = 2\pi$ for the example system.

It will come as no surprise given the Duffing oscillator discussion that stable and unstable manifolds play a critical role in deciphering the transport properties of Hill's vortex. Indeed the hyperbolic saddle points of Figure A.12(b) transform into saddle cycles for the periodic flow, and the spherical heteroclinic connection of the hyperbolic points disintegrates under perturbation forming a heteroclinic tangle. Segments of the intersecting stable and unstable manifolds of the saddle cycles are shown in Figure A.18. Note that manifolds along the line $x = 0$ of Figure A.12(b) remain intact, but have been omitted from Figure A.18 for clarity.

Remnants of the original vortex core remain in the perturbed case, and the manifolds can be seen as wrapping around this structure. Geometrically, the vortex core is once again isolated by invariant tori, although in the case of Hill's spherical vortex, these tori are now 3-surfaces in a 4D extended state space. A specially constructed Poincaré section, Figure A.19, confirms the presence of invariant tori. A couple of notes regarding Figure A.19 are in order. Each trajectory is assigned a color so that its behavior in the plane of section can be easily tracked. For example, the two inner-

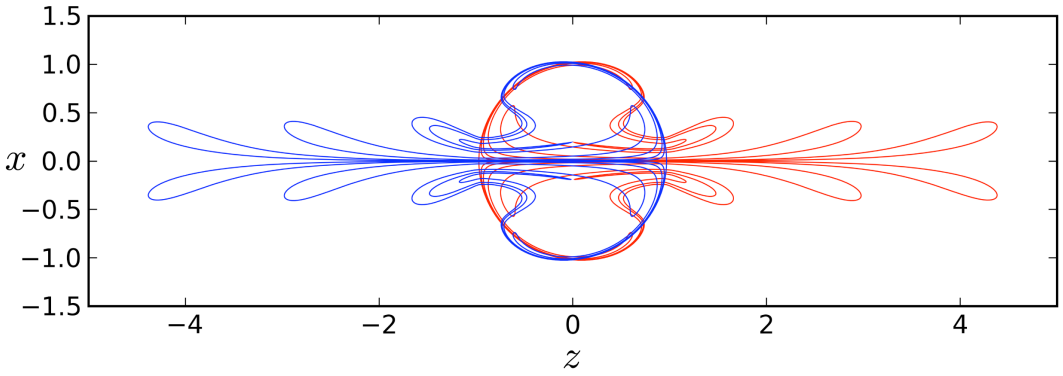

Figure A.18: Segments of the stable (red) and unstable (blue) manifolds for the perturbed Hill's spherical vortex.

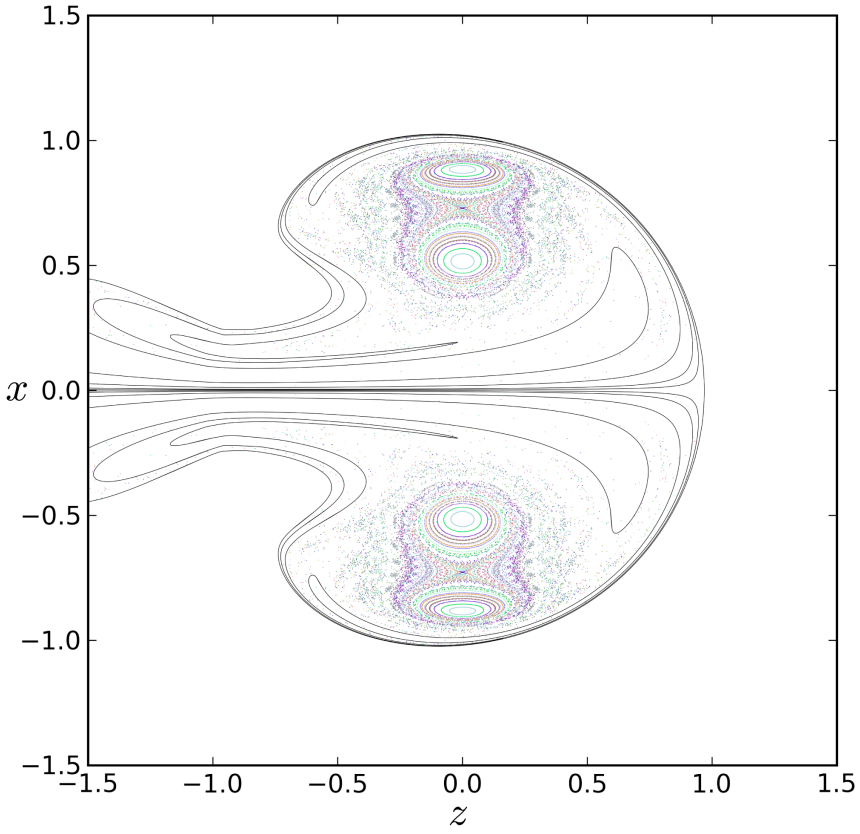

Figure A.19: Poincaré section for the perturbed Hill's spherical vortex. The unstable manifold (solid black line) of the leading edge saddle cycle has been included for reference. Trajectories have been colored for easy identification.

most gray circles in the $x < 0$ region are produced by the same trajectory. But the trajectory that produces the circles in the lower half of the figure does not produce those in the upper half — as the governing system of equations clearly indicates, there is no azimuthal component about the z-axis to the perturbed velocity field.

The stirring behavior demonstrated in Figure A.5 should now start to become clear. The red and blue sections of the blocks were initially released such that they were fully contained between the stable and unstable manifolds (ref. A.20). On the other hand, the cyan and orange tracers were situated just outside the perimeter of the stable manifold and consequently could never be brought into the stirring zone surrounding the vortex. For the cyan and orange tracers to enter the stirring zone, the impenetrable barrier formed by the manifolds would need to be breached. Since the perturbed Hill's vortex is periodic and assumed to exist for all time, the stable manifolds extend infinitely far along the positive z-axis encircling not just the red and blue tracers, but all mass that will ever be entrained into the stirring zone.

It might seem somewhat disturbing that we have classified invariant manifolds as impenetrable surfaces that somehow permit the red and blue fluid to move within the stirring zone but simultaneously prevent the same behavior for the cyan and orange. The means by which these structures transport material can be viewed in one of two ways. First, we can revert to thinking of the Hill's vortex as a non-autonomous system in which the now time-varying manifolds can stretch and deform as the flow evolves. The manifolds from this viewpoint are simply special material surfaces that move under a unsteady velocity field and have the property that any point along the manifold decays to some distinguished, parent hyperbolic trajectory in an infinite time limit. From the other perspective, we can continue to treat the system from an autonomous viewpoint in which case the compartments formed by the manifold intersections are like tubes transporting fluid through extended state space. Regardless, if material is to be brought into the stirring region, it must already be

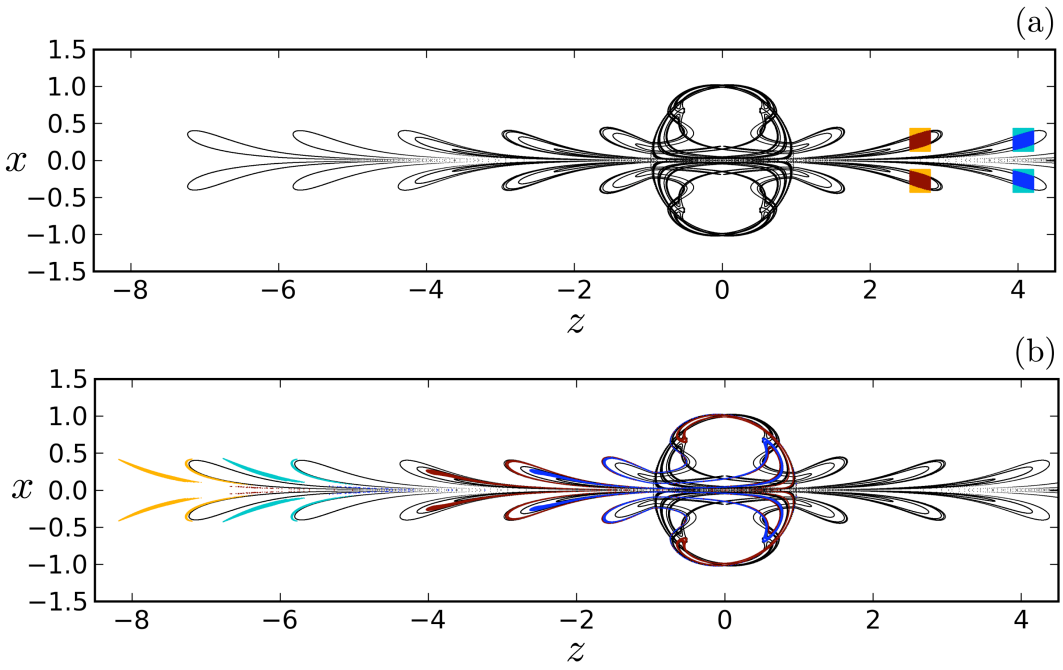

Figure A.20: Passive tracer blocks from Figure A.5 superimposed on the stable and unstable manifolds (black) of the saddle cycles for the perturbed Hill's spherical vortex at times (a) $t = 0$ and (b) $t = 8$.

encapsulated by the manifolds. These mechanisms by which fluid is entrained into a stirring zone can actually be put on a firm analytical footing for periodic flows by way of lobe dynamics, and the interested reader is referred to [29, 30, 22, 10, 40].

Although mentioned in a previous discussion of the Duffing oscillator, from Figure A.20(b) it should be quite clear that the evolved tracer blocks approximate the shape of the unstable manifold. This behavior is a standard characteristic of many flows. That is dye, smoke, or other flow visualizations generally take the shape of a hyperbolic trajectory's unstable manifold. A conceptual understanding for how the unstable manifold dominates visualization can be seen by way of Figure A.8(c). The straight, diagonal lines in the figure are actually the stable and unstable manifolds of the saddle point for this illustrated linear system. Notice how the non-manifold trajectories move away from the stable manifold with increasing time as though they are being repelled, but approach and decay to the unstable manifold. A block of

initial conditions released anywhere in Figure A.8(c) would be stretched out along the unstable manifold, and the degree to which the deformed block approximates the manifold would improve with time. As it turns out, this trajectory attracting property is used extensively in the literature (and within this document) to grow manifolds of hyperbolic trajectories. The general idea is that the manifold can be generated from either small segments of the manifold itself or from a ball of tracers centered about the hyperbolic point. When the tracers are evolved with the flow, they will be stretched out along the unstable manifold in forward time or the stable manifold in reverse time (see [16, 19, 25, 24, 28, 31]).[16]

A.6 Aperiodic systems and finite time dynamics

We have seen how non-autonomous systems can be converted to autonomous analogues by extending state space and discussed how such extended systems have no fixed points. During the non-autonomous to autonomous conversion process an additional differential equation of the form $\dot{q} = c$ is created, where q is the new state space variable accommodating the explicit time dependence of the corresponding non-autonomous system, and c is a constant. Consequently, for the resulting extended autonomous system given by $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, there is no value of $\boldsymbol{x}$ such that $\boldsymbol{f}(\boldsymbol{x}) = 0$ unless $c = 0$. In the case where $c = 0$, the original system is already autonomous, and there is no need for a transformation.

Although the general autonomous system may have no fixed points, such systems can still have hyperbolic structures with stable and unstable manifolds that play crucial roles in the observed dynamics of the system, as was seen via the forced Duffing oscillator and Hill's spherical vortex of the previous section. Locating the saddle cycles and associated manifolds for both of these example systems is greatly

[16] Do not forget that in reverse time, the stable manifold of hyperbolic trajectories behaves analogously to the unstable manifold in forward time. Consequently, the stable manifold can be grown by advecting the tracers backward in time.

facilitated by the periodic nature of the forcing as well as the assumption of infinite time existence. When we lose periodicity or infinite time existence, then finding distinguished hyperbolic trajectories and related manifolds, let alone characterizing the long term behavior of such structures, becomes extremely difficult. Unfortunately, aperiodicity and data availability on finite time intervals are very common in practice. Any numerical dataset generated from a computer model or extracted via physical measurements is by necessity only known for a finite time. Aperiodicity can arise from the analysis of transient phenomena or simply be a consequence of measuring a flow field for an insufficient time to detect underlying periodic behavior.

So how do we find these distinguished hyperbolic trajectories, like saddle cycles, in aperiodic finite time datasets? Several schemes have been developed that exploit known features of any given system. One such approach was actually used to approximately locate the saddle cycle in Figure A.14. The location of the saddle fixed point for the unperturbed Duffing system was known to lie at the origin, and an assumption was made that the perturbed saddle cycle would probably be nearby. Capitalizing on the attractive nature of the unstable manifold by releasing a small ball of passive tracers at the origin and advecting the tracers using the perturbed velocity field for several forcing periods, a segment of the unstable manifold was grown (ref. [37]). Repeating the process in backward time grows a section of the stable manifold. The resulting intersection of the two segments should be very close to the true location of the saddle cycle provided the growth time used was sufficiently long. Another technique searches for distinguished hyperbolic trajectories using instantaneous stagnation points of the non-autonomous vector field as starting points [19, 20, 23]. These instantaneous stagnation points are values of $\boldsymbol{x}$ for which $\boldsymbol{f}(\boldsymbol{x}, t) = 0$ in Eq. A.6, but instantaneous stagnation points of non-autonomous systems are not fixed points nor are they valid trajectories. Nevertheless, distinguished hyperbolic trajectories sometimes live in close proximity to instantaneous stagnation points, so starting a search

at these stagnation points can be productive [20, 6].

The method of locating distinguished hyperbolic trajectories which the author of this appendix has had the most success with, however, is a hybrid technique discussed in [6]. First approximate locations of the manifolds are computed using a finite time formulation of Lyapunov exponents (FTLE) [14, 15, 32]. Then better approximations to the manifolds are grown from the FTLE proxy, and the hyperbolic trajectory is taken as the primary intersection of the grown manifolds. Let us briefly examine how the technique works.

Imagine for a moment that we know the location of the stable and unstable manifold roots, where the roots are taken as the section of the manifolds that directly abut the distinguished hyperbolic trajectory. In this root section of the manifolds, the manifold trajectories are exponentially decaying to the parent hyperbolic trajectory as $t \to \pm\infty$. If we know an approximate location of the manifold roots, the intersection will roughly mark the location of the distinguished hyperbolic trajectory. It turns out that we can often estimate the manifold roots using a finite time version of Lyapunov exponents. The formulation for the finite time Lyapunov exponents (FTLE)[17] is identical to that of Eq. A.25, except we drop the infinite time limit to produce

$$s_i = \frac{1}{2T} \ln \lambda_i, \quad i = 1, \ldots, n \tag{A.41}$$

where the λ_i are still the eigenvalues of $\boldsymbol{M}$ (Eq. A.21), but T is now the time interval over which the FTLE have been computed.[18]

To see why computation of the FTLE field from several trajectories often provides an approximate location of the manifolds, reflect back on Figure A.8(c) and picture releasing two tracers very close together in some proximity of the stable manifold. Further assume that the trajectories are advected by the non-autonomous vector

[17] The FTLE is sometimes referred to as the direct Lyapunov exponent (DLE).

[18] In most of the literature *the* FTLE is taken to be the value of s computed from the largest, positive eigenvalue of $\boldsymbol{M}$.

field $\boldsymbol{f}(\boldsymbol{x}, t)$ for some time T_p.[19] If the two tracers are released on the same side of the stable manifold, it is conceivable that the tracers will move similarly and possibly stay close to each other. Now envision that the tracers are released again such that they straddle and are on opposite sides of the stable manifold. As this scenario evolves in forward time, the tracers will inevitably be carried toward the unstable manifold and then start to pull apart quickly. While the separation between the trajectories may remain small after a time T_p in the first case, we are guaranteed to see an increase in separation for the latter provided T_p is sufficiently large. Of course, measuring the rate of trajectory separation is exactly what the Lyapunov exponent captures. Therefore by releasing a large number of tracer pairs into a vector field and numerically computing FTLE values for each trajectory in forward time, we rightfully might expect to see a ridge of large FTLE values approximating the shape of the stable manifold. The process is equally applicable to the unstable manifold, provided the system is evolved in backward time.

While using the FTLE field to approximately locate the manifolds and the parent distinguished hyperbolic trajectory can be a very useful technique, the method does have limitations.[20] Perhaps the most important limitation is that ridges in the FTLE field do not always correspond to manifolds of a hyperbolic trajectory. Remember that the Lyapunov exponents of Section A.4 were constructed as an infinite time limit. The motivation for the infinite time limit was to ensure that we were seeing the true dynamics of the system by excluding short term behavioral excursions. When we disregard the infinite time limit, the FTLE field can identify spurious structures that an infinite time limit would filter out. A shear flow, for example, can produce a ridge in the FTLE field since trajectories on either side of the shear layer are clearly separating, but the shear layer itself is typically not a manifold of a hyperbolic trajectory. Other techniques must therefore be utilized in some cases to determine

[19] Or equivalently by the autonomous vector field $\boldsymbol{f}(\boldsymbol{x})$.

[20] For an excellent overview of FTLE limitations, see [6].

whether an FTLE ridge is approximating the manifold of a distinguished hyperbolic trajectory or merely identifying a region of high shear [15].[21]

Additionally, the FTLE field (or even a field of infinite time Lyapunov exponents) will not always identify manifolds even if they are present. The linear flow of A.8(c) which was used above to explain how the FTLE works turns out to be a case where Lyapunov exponents will not locate the manifolds. The reason is that all trajectories in the linear flow of A.8(c) have the exact same set of Lyapunov exponents. Consequently there will be no ridge of elevated finite (or infinite) time exponent values to mark the location of the manifolds for this flow, and other techniques must be used [19, 20, 23].

A.7 Summary

In the preceding discussion we have seen how fixed points or more general distinguished hyperbolic trajectories can have a significant impact on mass transport within fluid flows. While the fixed point or distinguished hyperbolic trajectory determines behavior in the near vicinity of the structure, the corresponding stable and unstable manifolds organize the flow field into dynamically distinct zones, govern mass exchange between the different regions, and have a profound effect on fluid stirring as well as the shape stirred fluid assumes during the process. The manner in which two nearby trajectories behave can be used to classify fixed points and other arbitrary trajectories or state space structures by way of linearization, with a specially constructed linearization providing Lyapunov exponents as a measure of the rate in which trajectories move together or apart. And although application of dynamical systems concepts to aperiodic and finite time systems presents additional challenges which are still being addressed by active research, existing notions such as Lyapunov

[21] Ridges of the FTLE field corresponding to hyperbolic material lines, as opposed to those arising from high shear, are often referred to as *Lagrangian coherent structures* (LCS) [32].

exponents can nevertheless yield important insight into system behavior in many cases.

A.8 References

[1] K. T. Alligood, T. D. Sauer, and J. A. Yorke. *Chaos: An Introduction to Dynamical Systems*. Springer, 1996.

[2] V. I. Arnol'd. Small denominators and problems of stability of motion in classical and celestial mechanics. *Russian Mathematical Surveys*, 18(6):85–191, 1963.

[3] J. Barrow-Green. *Poincare and the Three Body Problem*. American Mathematical Society, 1996.

[4] M. V. Berry. Regular and irregular motion. In S. Jorna, editor, *AIP Conference Proceedings*, volume 46, pages 16–120, 1978.

[5] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations*. John Wiley and Sons, 5th edition, 1992.

[6] M. Branicki and S. Wiggins. Finite-time Lagrangian transport analysis: stable and unstable manifolds of hyperbolic trajectories and finite-time Lyapunov exponents. *Nonlinear Processes in Geophysics*, 17(1):1–36, 2010.

[7] L. Dieci, R. D. Russell, and E. S. VanVleck. On the computation of Lyapunov exponents for continuous dynamical systems. *SIAM Journal on Numerical Analysis*, 34(1):402–423, Feb 1997.

[8] G. O. Fountain, D. V. Khakhar, I. Mezic, and J. M. Ottino. Chaotic mixing in a bounded three-dimensional flow. *Journal of Fluid Mechanics*, 417:265–301, Aug 2000.

[9] G. O. Fountain, D. V. Khakhar, and J. M. Ottino. Visualization of three-dimensional chaos. *Science*, 281(5377):683–686, Jul 1998.

[10] E. Franco, D. N. Pekarek, J. F. Peng, and J. O. Dabiri. Geometry of unsteady fluid transport during fluid-structure interactions. *Journal of Fluid Mechanics*, 589:125–145, Oct 2007.

[11] I. Goldhirsch, P. L. Sulem, and S. A. Orszag. Stability and Lyapunov stability of dynamic-systems - a differential approach and a numerical-method. *Physica D*, 27(3):311–337, Aug 1987.

[12] H. Goldstein. *Classical Mechanics*. Addison Wesley, 2nd edition, 1980.

[13] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 1983.

[14] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149(4):248–277, Mar 2001.

[15] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 14(6):1851–1861, Jun 2002.

[16] G. Haller and A. C. Poje. Finite time transport in aperiodic flows. *Physica D*, 119(3-4):352–380, Aug 1998.

[17] R. C. Hilborn. *Chaos and Nonlinear Dynamics*. Oxford University Press, 2nd edition, 2000.

[18] M. Hirsch, S. Smale, and R. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, 2nd edition, 2003.

[19] K. Ide, D. Small, and S. Wiggins. Distinguished hyperbolic trajectories in time-dependent fluid flows: analytical and computational approach for velocity fields defined as data sets. *Nonlinear Processes in Geophysics*, 9(3-4):237–263, May-Jul 2002.

[20] N. Ju, D. Small, and S. Wiggins. Existence and computation of hyperbolic trajectories of aperiodically time dependent vector fields and their approximations. *International Journal of Bifurcation and Chaos*, 13(6):1449–1457, Jun 2003.

[21] J. A. J. Madrid and A. M. Mancho. Distinguished trajectories in time dependent vector fields. *Chaos*, 19(1):013111, Mar 2009.

[22] N. Malhotra and S. Wiggins. Geometric structures, lobe dynamics, and Lagrangian transport in flows with aperiodic time-dependence, with applications to Rossby wave flow. *Journal of Nonlinear Science*, 8(4):401–456, Jul-Aug 1998.

[23] A. M. Mancho, D. Small, and S. Wiggins. Computation of hyperbolic trajectories and their stable and unstable manifolds for oceanographic flows represented as data sets. *Nonlinear Processes in Geophysics*, 11(1):17–33, 2004.

[24] A. M. Mancho, D. Small, S. Wiggins, and K. Ide. Computation of stable and unstable manifolds of hyperbolic trajectories in two-dimensional, aperiodically time-dependent vector fields. *Physica D*, 182(3-4):188–222, Aug 2003.

[25] P. D. Miller, C. K. R. T. Jones, A. M. Rogerson, and L. J. Pratt. Quantifying transport in numerically generated velocity fields. *Physica D*, 110(1-2):105–122, Dec 1997.

[26] V. I. Oseledec. A multiplicative ergodic theorem: Lyapunov characteristic numbers for dynamical systems. *Transactions of the Moscow Mathematical Society*, 19:197–231, 1968.

[27] J. M. Ottino. *The Kinematics of Mixing: Stretching, Chaos, and Transport*. Cambridge University Press, 1989.

[28] A. C. Poje and G. Haller. Geometry of cross-stream mixing in a double-gyre ocean model. *Journal of Physical Oceanography*, 29(8):1649–1665, Aug 1999.

[29] V. Rom-Kedar, A. Leonard, and S. Wiggins. An analytical study of transport, mixing and chaos in an unsteady vortical flow. *Journal of Fluid Mechanics*, 214:347–394, May 1990.

[30] V. Rom-Kedar and S. Wiggins. Transport in two-dimensional maps. *Archive for Rational Mechanics and Analysis*, 109(3):239–298, 1990.

[31] I. I. Rypina, M. G. Brown, and H. Kocak. Transport in an idealized three-gyre system with application to the adriatic sea. *Journal of Physical Oceanography*, 39(3):675–690, Mar 2009.

[32] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D*, 212(3-4):271–304, Dec 2005.

[33] G. Strang. *Linear Algebra and its Applications*. Saunders College Publishing, 3rd edition, 1988.

[34] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993.

[35] S. H. Strogatz. *Nonlinear Dynamics and Chaos*. Westview Press, 1994.

[36] K. Sundman. Mémoire sur le problème des trois corps. *Acta Mathematica*, 36:105–179, 1912.

[37] P. D. Swanson and J. M. Ottino. A comparative computational and experimental study of chaotic mixing of viscous fluids. *Journal of Fluid Mechanics*, 213:227–249, Apr 1990.

[38] M. J. Valtonen and H. Karttunen. *The Three-Body Problem*. Cambridge University Press, 2006.

[39] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer, 2nd edition, 2000.

[40] S. Wiggins. The dynamical systems approach to Lagrangian transport in oceanic flows. *Annual Review of Fluid Mechanics*, 37:295–328, 2005.

APPENDIX B

SPIVET

B.1 Overview

SPIVET (Stereoscopic Particle Image VElocimetry and Thermometry) is a Python package providing a series of tools for analyzing stereoscopic PIV image sequences of experimental fluid or particle flows. The functionality provided by the package serves four primary purposes:

1. Extraction of a displacement (or velocity) field from raw PIV images.

2. Extraction of temperature values from images taken of thermochromic liquid crystals (TLC's).

3. Passive tracer advection framework for Lagrangian transport studies and computation of finite-time Lyapunov exponent fields.

4. Aggregation and storage of processed results in a portable file format that is compatible with a variety of visualization software suites (*e.g.*, ParaView [1], and VisIt [3]).

SPIVET services are provided by a collection of Python modules as described in the Architecture section (Section B.3). At present, SPIVET does not have a graphical

user interface, but is instead used via the command line and Python scripts. Example scripts are provided to get the user moving in the correct direction.

The present document is meant to provide the user with a introductory perspective of SPIVET, its structure, and its capabilities. Detailed module and function documentation is provided in the code and can be retrieved using Python's `help` function or by reading the code itself. A bibliography is also provided at the root of the source tree.

B.2 Licensing

SPIVET is released under the terms of Version 2 of the GNU Public License (GPL). A copy of the license is provided in the `LICENSE.SPIVET` file at the root of the source tree.

B.3 Architecture

SPIVET is an aggregate Python package composed of four principal lower-level components: PivLIB, TlcLIB, FloLIB, and SPIVET `steps`. A graphical depiction of SPIVET's architecture is shown in Figure B.1.

The choice of Python as the primary language for SPIVET was motivated in large part by Python's:

- clear, compact, and easy to learn syntax,
- superb ability to glue disparate applications and libraries together into one cohesive whole,
- and ease of quickly visualizing data via directly callable packages such matplotlib[4] (also referred to as pylab) and VTK [2].

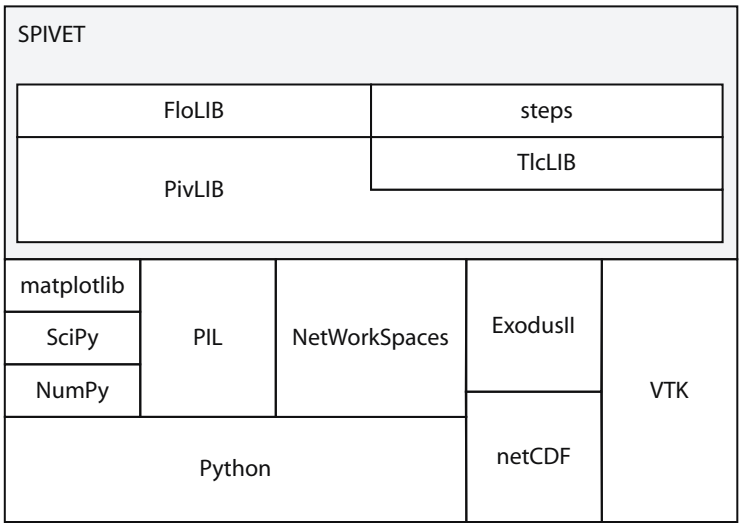

Figure B.1: Architectural overview of SPIVET. Primary dependencies on third party software are also shown (see Section B.7).

However, Python does have some drawbacks, the largest of which is shared with most other interpreted languages: namely, poor execution speed as compared to a compiled language such as C. Python's performance penalty is particularly acute when executing loops over large data arrays. In such loops, Python code can easily run an order of magnitude or more slower than comparable compiled code. There are two dominant methods for avoiding such overhead: vectorized statements and external C modules.

Often the cleanest, most compact way of negating performance penalties associated with Python loops is to use 'vectorized' statements which have an implicit loop that is often implemented in lower-level compiled code. To leverage this approach, much of SPIVET's internal workings are built around NumPy [6] `ndarrays` (n-dimensional array class). Using NumPy `ndarray` objects, two similarly shaped arrays can be added and results stored in a third array with the single Python statement: `arrayc = arraya + arrayb`. Not only is the preceding Python code substantially faster than an explicit Python `for` loop, but it is cleaner and easier to read. Nevertheless, there are cases where vectorized statements are difficult to formulate or

explicit loops are unavoidable. In these instances, SPIVET implements the necessary functionality in an external C module and subsequently wraps the module in Python.

SPIVET has been constructed to utilize parallel processing for extracting data from very large sequences of images[1]. Here again, Python has strengths and weaknesses. Threading of pure Python code is very problematic. A Python construct known as the Global Interpreter Lock (GIL) effectively serializes Python code executing within a single process[2]. To work around these limitations and still provide parallel processing for large image sets, SPIVET relies on the NetWorkSpaces [8] library to provide process-level parallelism and inter-process communication. Overall, SPIVET in cooperation with NetWorkSpaces functions very similar to a parallel program utilizing MPI [5]. NetWorkSpaces manages the spawning of individual worker processes on each processor of a networked grid of computers, and coordinates the exchange of data between worker processes. By using process-level concurrency, the serializing effect of the GIL is avoided since each process has its own, private GIL.

A good deal of effort has been spent to minimize, where possible, the virtual memory footprint of SPIVET. SPIVET's internal data structures (Section B.3.1) attempt to intelligently manage their utilization of virtual memory by temporarily storing dormant data to disk on a least recently used basis. This functionality frees virtual memory address space to be utilized for other purposes, and is of primary benefit to installations that still rely on a Python built for a 32-bit address space. The issue of virtual memory address space exhaustion will all but disappear once 64-bit operating systems and user-level applications (like Python) become standard.

The function of each of the four principal SPIVET sub-packages as shown in Figure B.1 is discussed in the following sub-sections. SPIVET's third-party dependencies are

[1]At present, SPIVET's use of parallel processing is limited to the reduction of raw images to flow field variables, as this is the most time consuming operation. Nonetheless, there are plenty of opportunities to utilize parallel programming in other parts of SPIVET (see Section B.8).

[2]Python places few constraints on external libraries written in a compiled language, and these libraries are free to employ threads, MPI [5], or any other concurrent execution techniques as long as the code does not call back into the Python API to interact with a Python object.

exodusII	Path containing the ExodusII library
__init__.py	PivLIB initialization module
exodusII.py	Python wrapper module for ExodusII library
pivcolor.py	Colormaps for use with pylab
pivdata.py	SPIVET data structures
pivir.py	Image registration functions
pivlibc.c	Numerically intensive C functions
pivlinalg.py	Streamlined LAPACK wrappers
pivof.py	Optical flow driver functions
pivpg.py	Photogrammetry functions
pivpgcal.py	Photogrammetric calibration functions
pivpickle.py	Pickling functions that compress data
pivpost.py	Filtering and spurious vector removal
pivsim.py	Ray tracing optical simulator
pivsimc.c	C functions for ray tracing
pivtpsc.c	C functions for thin-plate splines and shape contexts
pivutil.py	Miscellaneous utilities, including the reading and writing of images, used by many modules

Table B.1: PivLIB package contents.

covered more fully in Section B.7.

B.3.1 PivLIB

The PivLIB package provides the core algorithms for Particle Image Velocimetry and other essential SPIVET functionality. These services include camera calibration, extraction of displacement vectors from images, reconstruction of 3D displacements from 2D displacements (the stereoscopic aspect of PIV), post processing of displacement fields via filtering, reading and writing of images, a ray tracing simulator, and SPIVET data structures. The functionality of the PivLIB package is spread across the files as shown in Table B.1. A few notes follow.

SPIVET data structures

Internally, SPIVET stores non-image PIV data (*e.g.*, velocity) in a hierarchy of three, specialized, intelligent data structures that have been constructed to minimize the virtual memory footprint of SPIVET. The lowest level data structure is the `PIVVar` (a class derived from a NumPy `ndarray` [6]) which holds a single flow field variable (*e.g.*, temperature, velocity, viscous stress tensor, etc) along with the variable's name and units. Because the `PIVVar` class is derived from a NumPy `ndarray`, the full compliment of NumPy and SciPy functionality or any other framework that utilizes `ndarray`s can operate directly on a `PIVVar`. And just like `ndarray` objects, `PIVVar`s can be operands of standard arithmetic operators as shown in the following Python code snippet[3]

```
vara = varb + varc
```

The data stored in a `PIVVar` is ordered according to the underlying uniform grid of PIV cells (see Section B.4). `PIVVar` objects know how to temporarily store their contents to disk thereby freeing up virtual memory address space. The management of these 'active' and 'deactivated' `PIVVar` objects is provided by the SPIVET `PIVEpoch` container class.

In many experimental cases, multiple flow field variables are of interest. Indeed, SPIVET has been specifically crafted to extract velocity and temperature fields from PIV image sets, and the user may wish to compute further derived quantities. To this end, the `PIVEpoch` is a container class that stores the full set of flow variables which are valid at a given instant in time. The `PIVEpoch` class is derived from the standard Python dictionary and stores `PIVVar` objects by the variable's associated name. The `PIVEpoch` class autonomously maintains a cache of recently used `PIVVar`s,

[3]The user should note, however, that although the newly created variable `vara` is a valid `PIVVar`, it has no name or units. The units and name of a `PIVVar` created by an arithmetic operation is by default set to the string `'NOTSET'`. In such cases, the `PIVVar` method `setattr()` can be used to assign a meaningful name and units.

while transparently storing unused variables to disk.

The set of `PIVEpoch`s that describe the time evolution of a flow field are stored in the highest level container, the `PIVData` object. The `PIVData` class derives from the Python list and is a simple ordered container that can write its full contents to disk as an ExodusII [7] file. The `PIVData` object is also responsible for tracking cell size and mesh origin within SPIVET's internal world coordinate system (see Section B.4).

Permanent storage and data exchange

For permanent, external storage of flow field data, SPIVET uses the ExodusII library [7] which writes all data stored in a `PIVData` object to a single, portable `ex2` file that can be directly loaded into visualization packages such as VisIt [3] or ParaView [1]. The ExodusII file format has been developed by Sandia National Laboratories for storage of large, time-varying datasets expressed on unstructured grids. SPIVET represents its data on a uniform, cell-centered, structured mesh, so the unstructured mesh handling facilities of the ExodusII file format are not of particular benefit. Nevertheless, the portability, clean API, and ExodusII reliance on the NetCDF framework [11] for the underlying file structure are all significant advantages. An ExodusII file written by a SPIVET `PIVData` object is, for all intents and purposes, a disk-based incarnation of the `PIVData` object itself. Hence, SPIVET can easily 'reanimate' any `PIVData` object stored in and ExodusII file for additional processing at a later time.

There is one particular peculiarity of the ExodusII file format that the user should be aware of: variable names (with units appended) are currently limited to 32 characters. When SPIVET writes an ExodusII file, each ExodusII variable is assigned a name (limited to a maximum length of 32 characters) that is the concatenation of the corresponding `PIVVar` variable name, a space, and the `PIVVar` units enclosed in

__init__.py	TlcLIB initialization module
tlclibc.c	Numerically intensive C functions for TlcLIB
tlctc.py	Thermochromic functions that utilize an existing calibration
tlctccal.py	Thermochromic calibration
tlctf.py	Temperature extraction driver functions
tlcutil.py	Miscellaneous utility functions

Table B.2: TlcLIB package contents.

brackets (*i.e.,* `[]`).

Optical simulations

The `pivsim` module provides an optically correct ray tracer for experimental simulation. The functionality of the module is limited, but it has been used to do a full simulation of the entire PIV process, from photogrammetric calibration to a synthetic 'experiment.' In this manner, the module provides a simple means of investigating the relative importance of different factors on the aggregate PIV setup. Ray tracing simulations also permit accuracy analysis of PivLIB's PIV vector extraction algorithms in a well-controlled environment. However, the user should understand that such accuracy analyses often represent a floor since `pivsim` does not include errors from lens optics (e.g., spherical aberration, finite depth of field), camera noise, etc. These un-modeled errors are typically non-negligible and can actually be the dominant source.

B.3.2 TlcLIB

The TlcLIB package provides thermochromic liquid crystal (TLC) calibration and temperature field extraction facilities and is organized as shown in Table B.2.

B.3.3 SPIVET steps

The `spivet.steps` module and its helpers provide an object-oriented wrapper around the procedural functions contained in the PivLIB and TlcLIB packages that automate the process of converting raw images into desired flow field data. The `steps` module also separates user configuration data (contained in a 'recipe') from the function implementation details. Hence data processing scripts using `steps` are more compact since the script is only charged with configuring, initializing, and setting up the order in which the collection of steps are executed. Once a particular recipe is constructed, it can then be applied with limited or no modifications to a number of experimental datasets.

A recipe constructed from a collection of steps is the principal means for reducing raw image data into useable flow field variables such as velocity and temperature. Individual steps are provided to parse the full set of image files, partition the set of images for parallel processing, extract field variables, and post-process those field variables to remove spurious vectors or compute derived quantities (*e.g.*, vorticity). Users can also create their own derived steps which are stored in the `.spivet/usersteps` path under the user's home directory.

B.3.4 FloLIB

The set of FloLIB modules provide post-processing facilities for flow field data, with the primary functionality being: 1) computation of derivatives for field data stored in `PIVVars`, 2) spatial interpolation of field variables, and 3) Lagrangian transport analyses by way of passive tracer advection. FloLIB consists of the files shown in Table B.3.

__init__.py	FloLIB initialization module
floftle.py	Finite time Lyapunov exponent (FTLE) functions
floftlec.c	C library of numerically intensive FTLE functions
flotrace.py	Passive tracer functions
flotracec.py	C library of numerically intensive tracing functions
floutil.py	Generic utilities for derivatives and Runge-Kutta time-stepping
flovars.py	Functions for computing derived flow quantities
svcismat.h	Header file for tricubic interpolation

Table B.3: FloLIB package contents.

B.4 Coordinate system and array indexing

SPIVET generally orders data with the x-axis (i-index) varying fastest, followed by the y-axis (j-index), and finally the z-axis (k-index). Hence the Python statement

```
val = velocity[2,0,10,20]
```

where `velocity` is a `PIVVar`, assigns the x-component[4] of velocity in cell $(i, j, k) = (20, 10, 0)$ to the variable `val`. The only time this rule is broken is if SPIVET needs to pass information to an external library (*e.g.*, ExodusII) that requires a different ordering. In these special cases, the change is transparent to the user. Note that as with C, array indexing begins with zero in Python.

SPIVET employs several coordinate systems to convert raw images into useable flow field variables, however the three most important are undoubtedly local array coordinates, SPIVET's internal world coordinates, and `PIVData` coordinates. Local

[4]Note that the velocity components (first dimension of the `PIVVar`) are also ordered as z, y, x. This is another SPIVET convention that is *always* followed for *all* ordered data (`PIVVar`s or otherwise). A nine-component tensor stored in the `PIVVar atensor`, for example, will have its components ordered as $zz, zy, zx, yz, yy, yx, xz, xy, xx$. Hence `atensor[0,:,:,:]` will return an array of the zz-tensor component and `atensor[8,:,:,:]` will return the xx-component. Likewise a three-element vector, `tcrd`, containing the x, y, z-coordinates for a tracer would be stored in SPIVET as `tcrd = array([zcoord,ycoord,xcoord])`. When interacting with SPIVET, this convention must always be observed.

array coordinates are cell-centered and dimensionless with the coordinate system origin located at cell $(i, j) = (0, 0)$ for a 2D array (*e.g.*, an image) or $(i, j, k) = (0, 0, 0)$ for a 3D array (*e.g.*, a `PIVVar`). Local array coordinate axes point in the direction of increasing index.

Once photogrammetric calibration has been performed, SPIVET will project camera images back onto the plane of illumination (created by laser or light-sheet). This projection results in a dewarped representation of the camera image where all of the new 'world pixels' can be scaled by one single constant to represent distances in mm. SPIVET adopts a convention regarding the orientation of world coordinates that is common, but certainly not universal, in computer graphics and image processing applications. SPIVET takes the internal world coordinate system to be right-handed and to have its origin located at the upper left corner of the plane (or collection of planes in 3D) being imaged or viewed[5]. The internal world z-axis is perpendicular to this viewed plane, oriented such that z-coordinates increase away from the viewer (or camera). The internal world x- and y-axes are parallel to the local i- and j-axes, with x increasing to the viewer's right and$y$ increasing down. SPIVET is designed to process images taken of many planes within the flow field, where each plane is imaged by moving the light sheet and cameras as needed. SPIVET assumes this motion is parallel to the world z-axis. The origin of the world coordinate system is therefore located in the center of world pixel $(i, j) = (0, 0)$ (the upper left pixel) of the first plane imaged. SPIVET world coordinates have units of mm.

For flexibility, the `PIVData` coordinate origin does not have to be coincident with the SPIVET world coordinate origin. Since SPIVET is principally built around correlation particle image velocimetry (CPIV) techniques, extracted PIV displacement vectors represent an average displacement for a group of world pixels (also known as an interrogation window in the literature). Hence the cell-centered local array origin

[5]So when looking at a computer monitor, for example, the origin is the upper left corner of the screen using this convention.

of a `PIVVar` corresponds to a location in the center of the interrogation window. For obvious reasons, choosing the origin of the `PIVData` coordinate system to be coincident with the underlying local array origin is very beneficial. To this end, the `PIVData` `origin` member stores the offset between the origins of the SPIVET world system (centered in world pixel $(i, j) = (0, 0)$) and `PIVData` coordinate system (centered in `PIVData` cell $(i, j, k) = (0, 0, 0)$).

Although the origins of `PIVData` and SPIVET world coordinates can be offset, SPIVET does expect the particular coordinate axes to be parallel *and* point in the same direction. For the University of Michigan setup, the laboratory equipment sweeps the light sheet (and cameras) in a direction moving opposite the cameras' gaze. According to the definition of SPIVET world coordinates above, the light-sheet and camera are both moving in the negative z-direction. Given that SPIVET stores the extracted data from each plane in the order in which it was acquired, the `PIVData` local array k-axis points in the opposite direction of the world z-axis. The apparent mismatch is accounted for by another `PIVData` member, `cellsz`, which stores the cell size of the underlying `PIVData` mesh in units of mm. Values stored in `cellsz` can be negative, as is the case for the University of Michigan setup. `PIVData` cell centers can then be represented in valid SPIVET world coordinates by the pseudocode

```
cell_coords = cell_indices*pd.cellsz + pd.origin
```

where `pd` is taken to be a valid `PIVData` object.

B.5 Installation

SPIVET has been installed and tested on Linux and Mac OS X platforms. Some operations used within SPIVET are likely not portable to Windows machines. The installation procedure is detailed in the `README` file located in the root of the source tree.

B.6 Filesystem layout

The SPIVET source code is organized as shown in Figure B.2.

B.7 Dependencies

SPIVET requires the following software to be installed. Unless otherwise noted, the latest version available should work.

- **Python** SPIVET has not been tested against Python version 3.0 or above. SPIVET was written for and works with Python 2.5 and 2.6. Python 3.0 and above is not backward compatible with the Python 2.x series. As a result, running SPIVET in Python 3.x will undoubtedly require some changes to the SPIVET code. See `http://docs.python.org/dev/3.0/whatsnew/3.0.html` for more info on the differences between Python 2.x and Python 3.x.

- **NumPy** Provides scientific computation facilities and efficient handling of arrays [6].

- **SciPy** SciPy provides higher level scientific computation facilities that are built on top of NumPy [9].

- **PIL** Python Imaging Library. PIL provides facilities for reading and writing images of various formats [10].

- **netCDF** Low-level file format [11] upon which the ExodusII specification [7] is built. Users only need to install netCDF as the ExodusII library is provided as part of SPIVET.

- **VTK** Visualization toolkit that provides storage, visualization, and geometric data manipulation capabilities [2].

- **matplotlib** Python package providing 2D plotting capabilities [4].

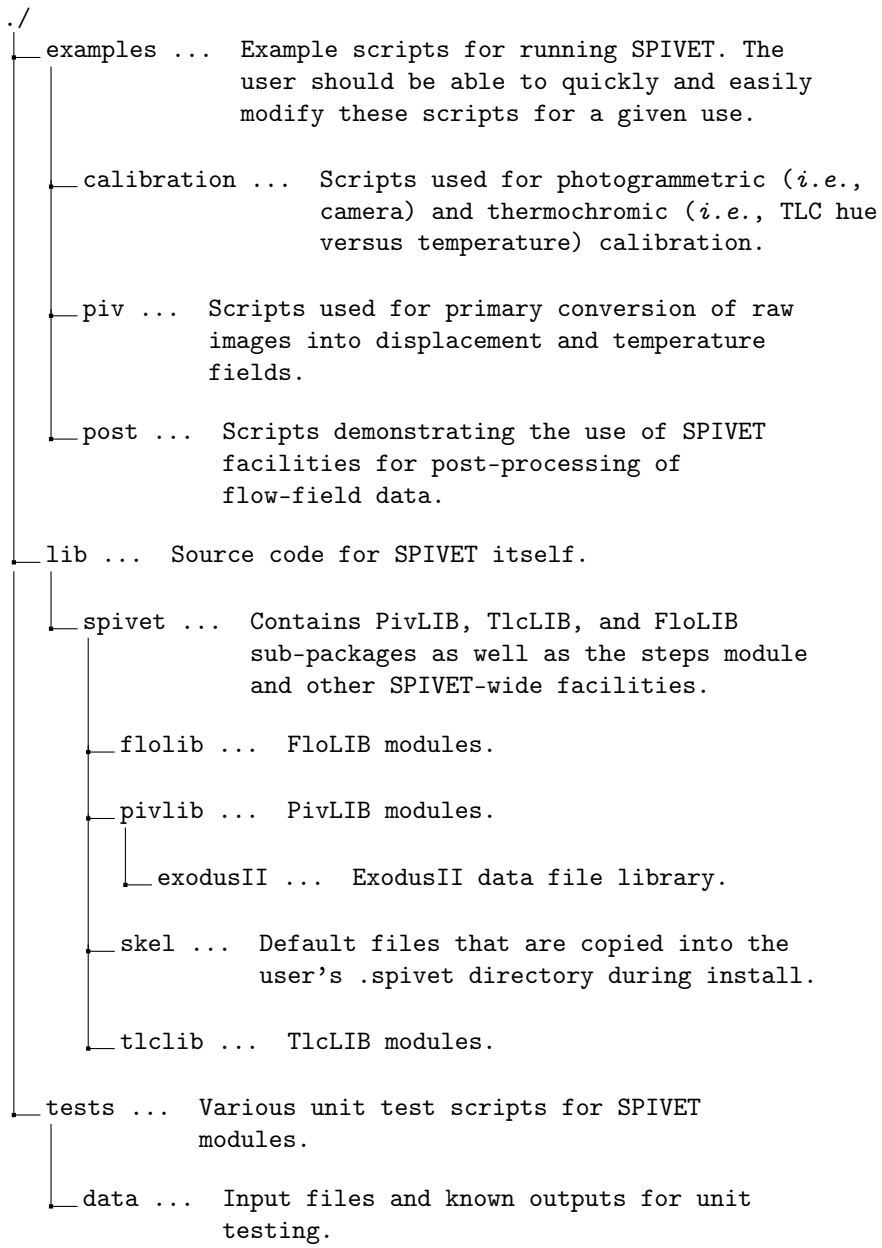

Figure B.2: SPIVET source code directory structure.

- **NetWorkSpaces** Provides the infrastructure for parallel processing [8]. NWS isn't required, but when processing large batches of images it's highly recommended. An NWS installation must consist of the client (installed on all worker nodes) and the server (on one machine only). Note that the server does not have to be a worker node.

B.8 Avenues for improvement

While SPIVET generally does a reasonable job at tasks for which it was designed, there are a number of opportunities for improvement. The more important deficiencies are discussed in this section.

As discussed in the overview, SPIVET is currently without a graphical user interface (GUI). SPIVET was not designed with real-time processing or user interaction in mind. Instead, SPIVET is a library predominantly geared toward batch processing of large quantities of images, extracting flow field variables from those images, and then post processing the results as needed. For most of SPIVET's intended uses, a GUI would seem to provide little or no benefit. However a GUI that manages the creation, organization, and configuration of SPIVET `steps` recipes (see Section B.3.3) would be quite valuable.

Significant improvement in execution speed and memory footprint could be realized by migrating some parts of the PivLIB image processing C code to single-precision arithmetic (or even integer arithmetic) and potentially vectorizing the corresponding code with SIMD instructions. External modules written in C or Fortran can use single- or double-precision arithmetic, and the NumPy `ndarray` class provides a mechanism for specifying whether a Python `ndarray` object holds single- or double-precision data. So facilities are available for working with single-precision data. Unfortunately, Python proper (with the exception of NumPy's `ndarray` class) is hardwired to use double-precision (*i.e.*, 64-bit) floating point arithmetic by de-

fault, so care must be taken to ensure that a Python coerced conversion from or to double-precision doesn't negate any single-precision performance benefits.

Having an optically accurate ray tracing facility like that of `pivsim`, can be extremely valuable in analyzing experimental errors, developing extensions to existing PIV algorithms, and upstream planning for new laboratory experiments. The functionality provided by the `pivsim` module, however, is currently very limited. Commercial optical design packages provide a much larger feature set and have substantially more robust computational geometry kernels. However, these commercial packages tend to be expensive and require a substantial learning curve due to the very large feature set provided. It is this author's opinion that investing in a commercial package could be a wise investment since the software would provide the PIV practitioner with the opportunity to design and analyze experiments that optimally leverage equipment capabilities. Nevertheless, the simulation facilities provided by `pivsim` can still be useful particularly if run-time performance is improved. In the current configuration, `pivsim` is implemented utilizing Python objects constructed from a mixture of Python and C code. Code that is called repeatedly during ray tracing is fully implemented in C, but still uses the Python object model and corresponding API. Unfortunately, interacting with Python objects has substantial overhead regardless of whether the object is implemented in Python or C. The expense is a consequence of Python's interpreted nature and the resulting way in which class data and function members are accessed (using a string search of a dictionary). The performance of `pivsim` can be improved substantially by eliminating all vestiges of Python and migrating the entire existing implementation to C++. Python could then be used to call into this self-sufficient library to setup the simulation and trigger the actual ray tracing, however all other functionality should be implemented independently of the Python API.

The passive tracer framework of FloLIB is under enormous strain from the perspectives of computational time and memory use. Advecting large numbers of passive

tracers is by its very nature a computationally expensive and memory intensive undertaking. The algorithms as currently implemented in FloLIB provide a high degree of flexibility for advecting tracers that can later be used for a variety of purposes, but execution time and memory constraints effectively limit FloLIB's passive tracing facilities to 2D applications. High-resolution 3D tracing operations will require the underlying algorithms to be more tailored to the specific end-use. As a case in point, consider the advection of tracers for the purpose of computing the finite-time Lyapunov exponent (FTLE) field. FloLIB's generic passive tracer functions are currently used to compute and store the full set of tracers at each time-step. The tracer field is then later re-processed by separate FTLE-specific algorithms. As far as FTLE computation is concerned, this decoupling of tracer advection from FTLE computation consumes a tremendous amount of both computation time (much of which is wasted writing and then re-reading the tracer field) and storage space. If the user's principal interest is the FTLE field, however, a small cluster of tracers could be followed in time, the desired FTLE field computed, and the tracer coordinates themselves discarded. This application-specific tailoring of passive tracing algorithms could significantly improve performance[6].

At present, SPIVET enforces the use of mm units for internal world coordinates. This can lead to convoluted combinations of units when computing derived quantities such as density. SPIVET should permit the user to specify arbitrary spatial units.

Although the majority of the most important functions have unit tests to help verify that SPIVET is operating correctly, not all parts of SPIVET have sufficient test cases. Implementing additional unit tests for under-tested or untested parts of the

[6]Tailoring of the code to end-use applications can be accomplished without necessitating the construction of a whole host of nearly identical, but slightly different tracing functions. The key is to re-architect the existing code into a more modular, object-oriented framework, with perhaps the producer-consumer design model used as guidance. Actually the current non-object-oriented configuration isn't far from this concept; the existing consumers simply don't intercept the produced data until it has been written to disk. Re-architecting the passive tracer code would also permit the code to be run in parallel (it is currently a serial implementation).

SPIVET framework would greatly improve the robustness of the code to downstream changes.

B.9 References

[1] Kitware, Inc. ParaView - Open Source Scientific Visualization. http://www.paraview.org.

[2] Kitware, Inc. VTK - The Visualization Toolkit. http://www.vtk.org.

[3] Lawrence Livermore National Laboratory. VisIt Visualization Tool. https://wci.llnl.gov/codes/visit.

[4] matplotlib. matplotlib: Python Plotting. http://matplotlib.sourceforge.net.

[5] MPI Forum. Message Passing Interface Forum. http://www.mpi-forum.org.

[6] NumPy. Scientific Computing Tools for Python - NumPy. http://numpy.scipy.org.

[7] Sandia National Laboratories. ExodusII. http://sourceforge.net/projects/exodusii.

[8] Scientific Computing Associates, Inc. NetWorkSpaces. http://www.lindaspaces.com/products/NWS_overview.html.

[9] SciPy. SciPy. http://numpy.scipy.org.

[10] Secret Labs AB. Python Imaging Library. http://www.pythonware.com/products/pil.

[11] Unidata. NetCDF (network Common Data Form). http://www.unidata.ucar.edu/software/netcdf.