Kate Royce

11/23/15

MA 53

Numerically Approximating Stable and Unstable Manifolds for Discrete Dynamical Systems in Two Dimensions

The study of chaotic dynamical systems has revolutionized the way computers are used in mathematics. Not only is the iteration of chaotic functions impossible to predict using computers,[[1]](#footnote-1) many features of chaotic systems are time-consuming and algorithmically difficult to model. Finding stable and unstable manifolds of saddle fixed points of a function is one such problem. A saddle fixed point of a map f has at least one attracting direction and at least one repelling direction (Figure 1). These directions are given by the eigenvalues of the local linearization matrix, the Jacobean, around the point; one eigenvector must be greater than 1 and one less than 1 for the fixed point to qualify as a saddle. The stable manifold is the set of initial points whose orbits under the function f converge to the saddle, and the unstable manifold is the set of initial points whose orbits under the inverse function, f-1, converge to the saddle. These sets are invariant under f and f-1, and, for two-dimensional maps, are always one-dimensional curves. Each fixed point of a 2-dimensional map can only have one stable and one unstable manifold.[[2]](#footnote-2)

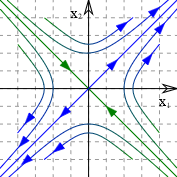


Figure 1: A saddle point with stable (green) and unstable (blue) manifolds[[3]](#footnote-3)

Many algorithms have been created to find these manifolds numerically, as they are impossible to discover analytically for nonlinear maps. This project will present some of the better algorithms for finding stable and unstable manifolds, discuss an algorithm that combines the best features of these methods, and implement this process using Matlab. For the sake of clarity, this algorithm will only solve 2-dimensional maps and must be given a fixed point of the map, along with its definition and inverse. Lastly, the correctness of the algorithm will be tested using examples from the literature, such as the Henon map and the Ikeda map.

Calculating a stable manifold is more difficult than calculating an unstable one, as most orbits follow the unstable manifold upon iteration. Most algorithms—including the one implemented here—approximate the stable manifold by calculating the inverse function’s unstable manifold, calculating backward iterations instead of forward. The most difficult aspect of graphing these manifolds, however, is that they can quickly diverge from the directions given by local linearization. The eigenvalues of the Jacobean matrix at the fixed point provide a framework for calculating manifolds. The eigenvector corresponding to the larger eigenvalue is the direction of the unstable manifold, and that of the smaller is the direction of the stable manifold. However, chaotic functions have sensitive dependence on initial conditions, meaning that after a few iterations these eigenvectors cannot predict the direction of the manifold, and many such functions have exponential separation rates between a point and its image. As a result, any algorithm which successfully graphs these manifolds must adapt to errors as they occur.

Krauskopf and Osinga (1998) present a way to calculate the next iteration using distance as a measure of accuracy. This algorithm starts with a cloud of points about the saddle fixed point and iterates them, but only graphs points within a set distance of a known point on the manifold. Repeating this process by creating a new “cloud” around the last known point refines the accuracy of the points that remain. Here, the speed of growth is determined by the curvature of the manifold rather than by the dynamics of the function (which could exponentially separate a point and its image). This approach, however, has the obvious drawback of requiring an invertible function. As a result, it is a poor fit for natural models or even maps like the “cat map”, which uses a modulus operation and is therefore not one-to-one. In addition, repeatedly operating on a cloud of points can make the algorithm too inefficient to be feasible for a large number of iterations.[[4]](#footnote-4)

A solution to this problem of inverse functions, if not time complexity, is the “search circle” algorithm described by England et al. (2004), which only requires the function and its saddle point as input. While this method calculates the unstable manifold in the same way as many others, it does not use the inverse function to compute the stable manifold. Instead, it uses the Jacobean to find the non-dominant eigenvectors, the initial directions of the stable manifold, and searches an arc of a given radius around the last known point on this manifold (initially just on an eigenvector) to find a point which maps onto the manifold under f. While time-consuming, this method is applicable to a much larger range of functions than an algorithm that requires an invertible function.[[5]](#footnote-5) This method controls resolution by only searching an arc of preset radius and angle for the next point.[[6]](#footnote-6)

The algorithm described here uses the function, under whose iterations the unstable manifold is plotted, and the inverse function, used to plot the stable manifold. That the algorithm requires an inverse function is not too large a problem, because one can use Newton’s method to find the inverse as in (7).[[7]](#footnote-7) It first calculates four points at a set distance from the fixed point p, and uses these points as the seeds of iteration. The algorithm iterates these points a fixed number of times, but keeps track of the distance between the points’ images and adds the midpoint of the line segment between them to the next iteration. Through this process, the line segments between points become a better approximation of the one-dimensional manifold. The algorithm terminates after a set number of iterations.

*while n is less than set number of iterations*

*plot both manifolds*

*for each ordered pair of points on stable manifold*

*if distance is greater than maximumDistance*

*add midpoint to stable manifold*

*end if*

*end for*

*iterate points on stable manifold*

*for each ordered pair of points on unstable manifold*

*if distance is greater than maximumDistance*

*add midpoint to unstable manifold*

*end if*

*end for*

*iterate points on unstable manifold*

*end while*

Code Fragment 1: Pseudocode describing the process of adding points to the manifold.

The algorithm presented here relies on the fact that the unstable manifold is the attractor for points near p under f, and the stable manifold is the attractor under f-1. It resolves the problem of exponential growth of iterations by adaptively adding points on the image of the manifold to be iterated. When the distance between two points grows larger than a preset distance (here 0.05), the algorithm adds the midpoint of the line segment between these points and iterates that point on the next iteration. While a smaller distance such as 0.01 leads to more accuracy, it makes the algorithm unfeasible to run, and distances larger than 0.15 between consecutive points cause the algorithm to ignore sections of the manifold. Using a distance of 0.05, the algorithm is able to approximate the curves of the manifold even though it only starts with a small number of points to be iterated. While small errors, such as nonexistent homoclinic points, are present at low iterations, they are resolved at higher iterations (Figure 3). As shown in Figures 3-7 (compare with (8) for accuracy), it produces a good approximation of stable (green) and unstable (blue) manifolds for the maps defined to the left in approximately 10 iterations.

The maps used to test the algorithm:

1. Henon map: (a- x2-by, x)
2. Ikeda map: (R + C2(xcos(t)-ysin(t)), C2 (xsin(t)+ycos(t)), t = (C1-C3)/(1+x2+y2)
3. Cat map: (2x + y, x +y)
4. Linear map: (3x, y/2)

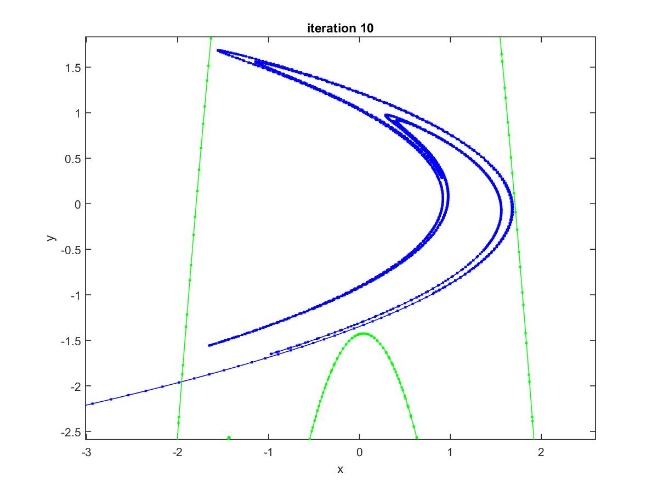
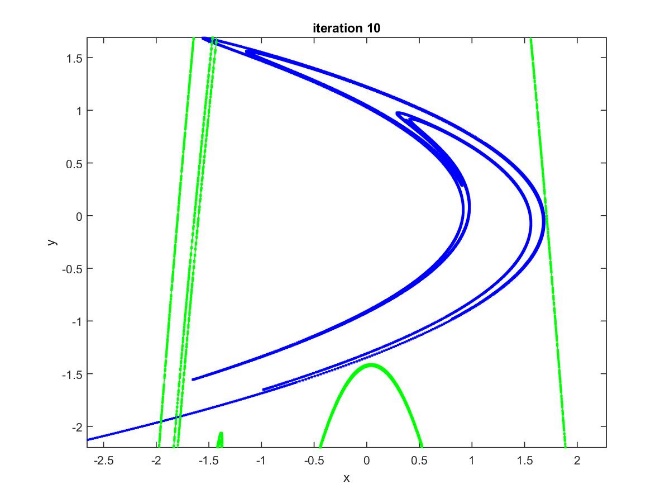


Figure 3: The stable (green) and unstable (blue) manifolds of the Henon map with a = 1.28,b = -0.3. While the manifolds should not cross for these parameters, the algorithm produces a homoclinic point at the upper left at iteration 8. At iteration 10, however, these mistakes have been corrected. These images represent the algorithm correcting itself over time.

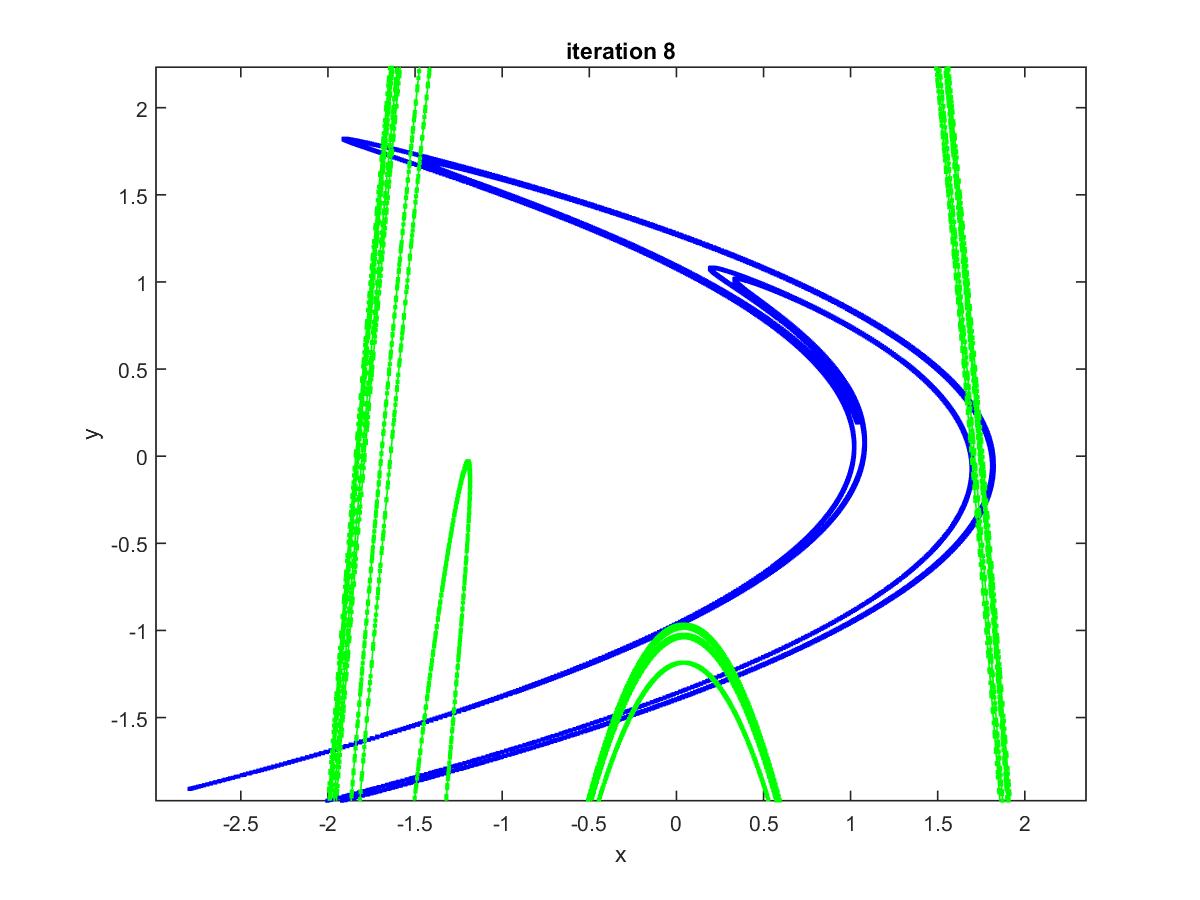


Figure 4: The stable and unstable manifolds of the Henon map with a = 1.4, b = -0.3. These parameters lead to chaos, as shown by the fact that the manifolds cross at several homoclinic points, and fractal formations.

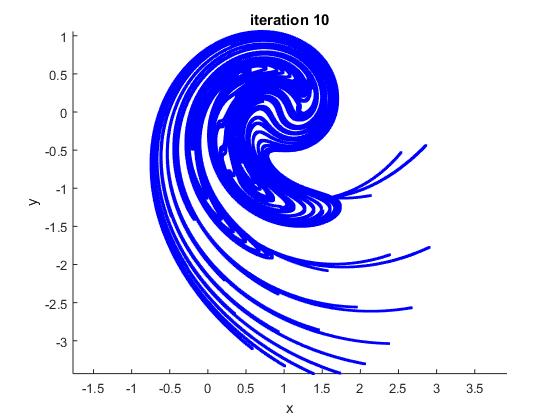


Figure 5: The unstable manifold of the Ikeda map[[8]](#footnote-8) with

r = 1, C1 = 0.4, C2 = 0.9, C3 = 6. The more linear shapes on the right are the first iteration of the four seeds, which have not yet found the true manifold.

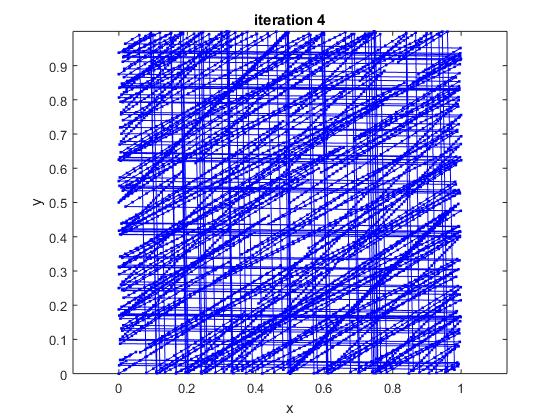


Figure 6: The unstable manifold of the “cat map”, which uses a modulus operation and does not have an inverse. The manifolds of this map are dense in the unit square, and the algorithm was stopped after the 4th iteration to preserve detail (at higher iterations, it is just a square of blue).

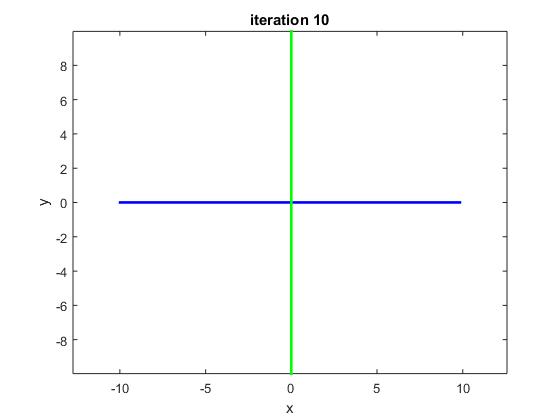


Figure 7: A linear map, with the stable and unstable manifolds in the direction of the smaller and larger eigenvectors ((1, 0) and (0, 1) respectively).

There are drawbacks to this system; for instance, the manifold’s shape varies between iterations when the number of iterations is small, producing errors such as the homoclinic point in Figure 2. This algorithm only uses 4 points as the initial seeds of the manifold, but still is expensive in terms of time. It depends on both the number of iterations (n) and the maximum distance (m) allowed between points, and grows exponentially as this distance shrinks. In the worst case, when a point must be added between every point already on the manifold, it requires O(2n/m) time. However, the algorithm only requires O(n) space to store the points on the stable and unstable manifolds, as the number of these points is bounded by n multiplied by the rate of separation (the Lyapunov exponent).

A subtle flaw of this algorithm is its ability to choose the seeds for the original iteration. Currently, it starts with four points a preset distance and direction from the fixed point, whatever it may be. These points could be chosen so that they lie directly on the opposite manifold, in which case the algorithm would not be able to accurately graph the correct one. Although unlikely, this problem can be fixed by examining the local linearization of the fixed point, the Jacobean derivative matrix, and choosing seeds on the correct eigenvector. This algorithm, however, avoids the complexity inherent in this method by choosing four points in orthogonal directions from the fixed point. It is impossible that the opposite manifold contains all four of these points.

The algorithm itself can be improved as well. A more efficient implementation of this algorithm would not need to store all the points it uses in one array. Perhaps a better implementation would delete points from the list to be iterated when it knows it has already computed the iterate of these points, or replace points with their iterate to save space. The implementation presented here continues to store every point in its list to be iterated over and over, thus decreasing the efficiency of the program as a whole.

Guckenheimer (1999) suggests additional factors that an analysis of manifold behavior should consider.[[9]](#footnote-9) Firstly, it would be interesting to graph the manifolds of a saddle-like periodic orbit, rather than just a fixed point. Secondly, this algorithm may be extended to Rm, where manifolds are objects in R2 or higher, by considering the Lyapunov exponents and computing the Jacobian derivative matrix around the saddle fixed point in question, as is done in (Haller, 2000).[[10]](#footnote-10) However, graphing manifolds of fixed points of maps in R3 or higher dimensions may not be easy to compute by drawing lines between points. The approach presented here relies on the fact that the manifold is a 1-dimensional object, which cannot be assumed for higher-dimensional maps. An algorithm that could graph 2D or 3D manifolds would have to consider 3 or 4 eigenvector directions when it chooses seeds for iteration. Graphing the manifolds of 2-dimensional maps provides a sample of the computational complexity involved in the study of dynamical systems, and is thus an indispensable starting step when analyzing these systems.

**References**

1. Devaney, R.L. 1992. *A First Course in Chaotic Dynamical Systems.* Westview Press.
2. Smith, H.L. 2011. *Stable and Unstable Manifolds for Planar Dynamical Systems.* Arizona State University.
3. Ghaziani, R.K., Govaerts, W., Kuznetsov, Yu.A.; Meijer, H.E. 2009. Numerical Continuation of Connecting Orbits of Maps in MATLAB. *Journal of Difference Equations and Applications.* 15 (8-9), pp 1-31.
4. Krauskopf, B. and Osinga, H. 1998. Growing 1D and quasi-2D unstable manifolds of maps. *J. Comput. Phys.,* 146: 404-419.
5. England, J.P., Krauskopf, B., Osinga, H.M., Computing one-dimensional stable manifolds and stable sets of planar maps without the inverse. *SIAM J. Applied Dynamical Systems* 3 (2004),161-190.
6. Kostelich, E.J., Yorke, J.A., You, Z. 1996. Plotting stable manifolds: Error estimates and noninvertible maps. *Phys. D*, 93, pp 210-222.
7. Ikeda, K. 1979. Multiple-valued Stationary State and its Instability of the Transmitted Light by a Ring Cavity System, *Opt. Commun.* 30 257-261.
8. Alligood, KT; Sauer, TD; Yorke, JA. 1996. *Chaos: An Introduction to Dynamical Systems.* Springer-Verlag New York, Inc; pp 83, 202.
9. Guckenheimer, J. 1999. Numerical analysis of dynamical systems. Mathematics Department, Cornell University, Ithaca, New York.
10. Haller, G. 2000. Finding finite-time invariant manifolds in two-dimensional velocity fields. Division of Applied Mathematics, Brown University, Providence, Rhode Island. American Institute of Physics.

1. Devaney, RL. 1992. *A First Course in Chaotic Dynamical Systems.* Westview Press. [↑](#footnote-ref-1)
2. Smith,HL. 2011. Stable and Unstable Manifolds for Planar Dynamical Systems. [↑](#footnote-ref-2)
3. "Phase Portrait Sadle" by Podshumok. Own work. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Phase\_Portrait\_Sadle.svg#/media/File:Phase\_Portrait\_Sadle.svg [↑](#footnote-ref-3)
4. Ghaziani, RK, Govaerts, W, Kuznetsov, Yu.A; Meijer, HE. 2009. Numerical Continuation of Connecting Orbits of Maps in MATLAB. *Journal of Difference Equations and Applications.* 15 (8-9), pp 1-31. [↑](#footnote-ref-4)
5. Bernd Krauskopf and Hinke Osinga, Growing 1D and quasi-2D unstable manifolds of maps. *J. Comput. Phys*., 146: 404-419, 1998. [↑](#footnote-ref-5)
6. England,J.P., Krauskopf, B.,Osinga,H.M., Computing one-dimensional stable manifolds and stable sets of planar maps without the inverse, *SIAM J. Applied Dynamical Systems* 3 (2004),161-190. [↑](#footnote-ref-6)
7. EJ Kostelich, JA Yorke, Z You. Plotting stable manifolds: Error estimates and noninvertible maps. *Phys. D,* 93 (1996), pp 210-222.

   [↑](#footnote-ref-7)
8. K.Ikeda, Multiple-valued Stationary State and its Instability of the Transmitted Light by a Ring Cavity System, *Opt. Commun*. 30 257-261 (1979). [↑](#footnote-ref-8)
9. Guckenheimer, J. 1999. *Numerical analysis of dynamical systems*. Mathematics Department, Cornell University, Ithaca, New York. [↑](#footnote-ref-9)
10. Haller, G. 2000. *Finding finite-time invariant manifolds in two-dimensional velocity fields*. Division of Applied Mathematics, Brown University, Providence, Rhode Island. American Institute of Physics. [↑](#footnote-ref-10)